Ontologie-basierte Text Interpretation für semantische Referenzierung

DIPLOMARBEIT

Im Fach Informatik-Ingenieurwesen

Eingereicht von

Qiang Li
Matrikelnummer: 24053
Telefon: 040 2384 2655
Email: qiang.li@tu-harburg.de

Erstgutachter: Prof. Dr. rer. nat. Ralf Möller
Zweitgutachter: Prof. Dr.-Ing. Günther Pawellek
Betreuer: MSc. Atila Kaya

Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Hamburg, 15. Dezember 2008

Unterschrift

Danksagung

An dieser Stelle möchte ich mich bei all jenen bedanken, die direkt oder indirekt zum Entstehen dieser Diplomarbeit beigetragen und mich auf verschiedenste Weise in deren Verlauf unterstützt haben.

Ein besonderer Dank gilt meinem Erstgutcher Herrn Prof. Dr. rer. nat. Ralf Möller für die Überlassung dieses interessante Themas. Auch meinem Zweitgutachter Herrn Prof. Dr.-Ing. Günther Pawellek gilt an dieser Stelle mein Dank.

Für die Bereitschaft, die vorliegende Arbeit über die gesamte Zeit ausführlich zu betreuen und mir zu jeder Zeit mit Rat und Tat zur Seite zu stehen, sowie für unzählige Tipps und fruchtbare Diskussionen danke ich besonders Herrn MSc. Atila Kaya. Ebenfalls bedanken möchte ich mich bei Dr. Lars Mühlenhoff, der die klinischen Studien als Testdokumente für diese Arbeit sorgfältig ausgewählt und mir Grundkenntnisse über diesen Kontexte vermittelt hat.

Für die grammatikalische und syntaktische Durchsicht bedanke ich mich herzlich bei Herrn MA Florian Krins. Ebenso möchte ich mich an dieser Stelle herzlich bei meinen Eltern, bei meiner Freundin für deren Unterstützung in jeglicher Hinsicht während der Dauer der Diplomarbeit und des Studiums bedanken.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

In jeder wissenschaftlichen Arbeit werden mehr oder minder ausgiebig die Gedanken, Ergebnisse, Daten, Materialien usw. anderer Veröffentlichungen herangezogen, verglichen, kritisch bewertet und als Grundlage eigener weiterführender Studien benutzt. Die Auseinandersetzung mit den Erkenntnissen anderer bildet die Voraussetzung, mit der eigenen Arbeit einen sinnvollen Beitrag zum Erkenntnisfortschritt leisten zu können. Eventuell entsteht dann eine neue Veröffentlichung. Die Auswertung fremden Wissens ist also nicht nur fruchtbar, sondern sogar notwendig, um Irrtümer und Doppelarbeit zu vermeiden und noch offene Fragestellungen zu erkennen. Allerdings gehört es zu den wesentlichen Grundprinzipien der Wissenschaftlichkeit, alle benutzten Quellen zu belegen – ihre Herkunft anzugeben – , damit sie in ihrem primären Kontext eingesehen und überprüft werden können [19].

Aber der Aufwand, Zitate korrekt zu erstellen ist sehr groß. Um die Quellen nach den Zitierrichtlinien richtig anzugeben, braucht man zahlreiche zusätzliche Meta-Informationen der benutzten Quellen, zum Beispiel Vor- und Nachnamen des Verfassers, Auflage, Verlagsort, Verlagsjahr usw. Wenn diese Informationen nicht direkt vorhanden sind, dann bedeutet es für den Autor eine lange Suchaktion. Nach dem Suchen braucht er noch Zeit, diese Meta-Informationen im richtigen Format und an der richtigen Stelle einzutragen.

1.1 Motivation

Klinische Studien bezeichnen die wissenschaftlichen Arbeit im medizinischen Bereich. Jährlich werden weltweit zahlreiche klinische Studien veröffentlicht. In einer klinischen Studie wird der Einfluss einer medizinischen Behandlung auf den Verlauf einer Krankheit in einem kontrollierten experimentellen Umfeld am Menschen erforscht. Die Studie von Arzneimitteln ist die gängigste Form der klinischen Studie. In Deutschland gibt es für solche klinischen Studien auch eine gesetzliche Definition.

> *[Die] Klinische Prüfung* bei Menschen ist jede am Menschen durchgeführte Untersuchung, die dazu bestimmt ist, klinische oder pharmakologische Wirkungen von Arzneimitteln zu erforschen oder nachzuweisen oder Nebenwirkungen festzustellen oder die Resorption, die Verteilung, den Stoffwechsel oder die Ausscheidung zu untersuchen, mit dem Ziel, sich von der Unbedenklichkeit oder Wirksamkeit der Arzneimittel zu überzeugen. (§4 Abs. 23 AMG[1])

Im Rahmen der Enbtwicklung eines neuen Therapieansatzes stellen die klinischen Studien den letzten Schritt in der Entwicklung dar. Sie sind ein wichtiger Bestandteil der Pharmaforschung [28].

Auf Grund der großen Anzahl der klinischen Studien ist die Nutzung dieser Studien und anderer medizinischer Dokumente für die meinsten Mitarbeiter aus dem medizinischen und pharmakologischen Bereich sehr ineffizient. Eine Studie [27] der Uni-Klinik Freiburg lautet, dass Daten in Kliniken zu 30% nicht dort sind, wo sie benötigt werden. Medizinische Dokumente sind zum Teil nicht lesbar, unvollständig, ungeordnet oder zu groß. Ca.5% aller ärztlichen Verrichtungen erfolgen ohne Kenntnis der für die Verrichtung notwendigen Informationen. Ungefähr 8,5% ihrer Arbeit verbringen Krankenhaus-Mitarbeiter täglich mit der Suche nach Daten.

[1] Gesetz über den Verkehr mit Arzneimitteln, kurz Arzneimittelgesetz

1.2 Zielsetzung dieser Arbeit

Deshalb wird ein kleines Anwendungsszenario aus Bereich des Gesundheitswesens für diese Arbeit ausgewählt. Mit dieser Arbeit wollen wir Ihnen eine möglichst einfache und komfortable Lösung präsentieren, damit Sie einen Text per semantischer Suche schnell finden und die gefundenen Dokumente direkt als Referenz in Ihrem Text einfügen können.

In dieser Arbeit wollen wir ein System entwerfen. Dieses System kann das Wissen automatisch aus Textdokumenten extrahieren und in eine Wissensbasis für die weitere Anwendung bereitstellen. Eine integrierte Komponente in ein Textverarbeitungsprogramm bietet der Anwender als Zugang zu dieser Wissensbasis.

1.3 Gliederung dieser Arbeit

Zunächst werden die Informationsextraktion und Beschreibungslogiken als Grundlagen dieser Arbeit in Kapitel 2 vorgestellt. Das Kapitel 4 beginnt mit einer Beschreibung unseres Anwendungsszenarios und der Beispieltexte. Durch die Analyse dieser Informationen entwerfen wir eine Systemarchitektur für die gesamte Arbeit. Am Ende des Kapitels 4 wird der Entwurf der Benutzeroberfläche mit der Kenntnis der Anwendungsanalyse dargestellt. Die wichtigen Prozessphasen in der Implementierung des Systems werden in Kapitel detailiert beschrieben.

Kapitel 2

Grundlagen

Als Grundlagen werden zahlreiche Grundkonzepte in dieser Arbeit verwendet. Für das Verständnis der weiteren Kapitel werden diese Grundlagen der Informationsextraktion und Beschreibungslogiken kurz dargestellt. Dieses Kapitel gibt einen kurzen Überblick über verwendete Begriffe.

Dieser Überblick erhebt keinen Anspruch auf Vollständigkeit.

2.1 Grundlagen der Informationsextraktion

Bevor Computer Ihnen einen inhaltlichen relevanten Text als Referenz geben können, müssen die Informationen in dem Text vom Computer verarbeitet werden. Vergleichsweise kann der Computer viel schneller und präziser strukturierten Informationen bearbeiten als wir Menschen. Aber im Gegensatz zur strukturierten Information ist ein Text normalerweise in natursprachlicher Form. Deshalb brauchen wir hier die Technik von Informationsextraktion um die Informationen aus Text zu gewinnen.

Wir stellen zunächst die Informationsextraktion vor. Dann werden wir die typische Aufgaben und Aufbauen eines Informationsextraktion-Systems kennenlernen. Am Ende dieses Kapitels zeigen wir Ihnen kurz GATE. Es ist ein Softwarewerkzeug, mit dem man viel typischen Informationsextraktionsaufgaben lösen kann.

2.1.1 Einführung

Informationsextraktion [12] (Information Extraction, IE) ist die Identifikation von Instanzen einer bestimmten Klassen von Ereignissen oder Beziehungen aus Texten in natürlicher Sprach, und die Extraktion der relevanten Argumentwerte des Ereignisses oder der Beziehung in vordefinierte Schablonen (Templates). Informationsextraktion involviert die Kreation von strukturierter Repräsentation in die ausgewählten Informationen aus dem Text.

Informationsextraktion ist eine Technologie. Sie ist aus Perspektive des Anwenders die Information zum Betrachten. Sie dient nicht nur dazu, dem Benutzer mitzuteilen, welche Dokumente er lesen soll, sondern sie schafft auch, Kenntnisse bezüglich einer in Vorhinein definierten Domäne zu gewinnen. Das Ergebnis von Informationsextraktion ist eine Menge von Fakten (Facts), die bestimmte Ereignisse, Entitäten oder Beziehungen entspricht. Fakten bedeuten hier die Relationen zwischen zwei oder mehrere Entitäten. Bei Informationsextraktion entsteht eine Verbindung zwischen extrahierenden Informationen und originalen Dokumenten. Diese Verbindung erlaubt dem Anwender, auf gesuchte Inhalte Bezug zu nehmen. Trotz dieser großen Ähnlichkeit unterscheidet sich die Informationsextraktion von der Informationswiedergewinnung (Information Retrieval, IR).

Es gibt wesentliche Unterschiede zwischen Informationsextraktion und Informationswiedergewinnung. [8].

- Ein IR-System findet den relevanten Text und zeigt ihn dem Benutzer.
- Eine IE-Anwendung analysiert den Text und zeigt nur bestimmte Informationen an. Es sind die Informationen, die für den Benutzer tatsächlich von Interesse sind.

Das Ergebnis von Informationsextraktion kann von einer IR-Applikation weiter benutzt werden. Eine Internet-Suchmaschine wie Google beispielsweise benutzt solche Ergebnisse, um Indexe zu erstellen.

Informationsextraktion ist auch nicht die Verständigung von natürlicher Sprache (Natural Language Understanding, NLU). Die klare Eigenschaften trennt sie von

NLU und anderen Technologien der Verarbeitung natürlicher Sprache (Natural Language Processing, NLP). IE Aufgaben werden in einer abgrenzten Domäne definiert. Deshalb sind die interessierenden Objekte auch eingeschränkt. Darüber hinaus konzentriert sich eine IE Aufgabe normalerweise auf ein bestimmtes, vorher definierendes Szenario innerhalb einer Domäne. Deswegen sind die Ereignisse und Beziehungen, an denen der Benutzer Interesse hat, auch begrenzt. Im Gegensatz zum NLU, IE kann man mit genauen Bewertungskriterien den Erfolg benoten. Die Evaluationskriterien werden wir später (siehe 2.1.3) besprechen.

Es gibt zwei grundlegende Ansätze für die Gestaltung von IE-Systemen. Der Erste ist ein alter und weiter verbreiterter Ansatz von Wissens-Ingenieurwesen (Knowledge Engineering Approach), der Zweite ist der Ansatz des automatischen maschinellen Lernens(Automatically Trainable Systems).

Der klare Vorteil des Ansatzes des Wissens-Ingenieurwesens besteht in den guten Leistungen. Hier braucht man eine/n ExpertertIn, der gute Kenntnisse über die Anwendungsdomäne und das Design eines IE-Systems haben sollte. Die entscheidenden Regeln für die Extraktion der gewünschten Informationen werden von diesem Experten definiert. Es wird auch eine Textsammlung aus relevanter Domäne für ihn bereitgestellt. Das Wissen und die Intuition, die dieser Experte besitzt, wird in der Regel ins System hereinfließen. Deshalb spielt dieser Fachmann eine entscheidende Rolle.

Der Prozess beim Ansatz des Wissens-Ingenieurwesen ist ein iterativer Vorgang. Durch jeden Iterativ werden die Regeln verwendet und die Ergebnisse verändert. Deswegen ist solcher Ansatz auch Kostenintensiv.

Gegenüber dem Ansatz von Wissens-Ingenieurwesen benötigt der Ansatz vom maschinellen Lernen keinen Experten. Stattdessen ist hier ein annotierendes domänspezifisches Korpus die Grundlage eines IE-Systems. Benötigt wird eine Person mit ausreichenden Kenntnissen über diese Domäne und die entsprechenden Annotationsaufgaben. Nach der Generierung vom annotierenden Korpus beginnt die Selbstlernphase der Maschine. Dann kann das System das Wissen, welches aus der annotierenden Korpus gewonnen wird, an neuen Texten verwenden, um die gewünschten Informationen zu extrahieren.

Für einige IE-Aufgabetypen wie „Named Entity Recognition“ ist der Prozess sehr

einfach. Aber bei komplexen Aufgaben kann die Annotation lästig sein. Die Schwierigkeiten solcher Annotationsprozesse wachsen mit der Komplexität der Aufgaben, die in IE-System erfüllt werden müssen. Im Extremfall kann es überschüssige Schwierigkeiten bei manueller Regelgenerierung geben.

Es ist unmöglich zu sagen, dass einer der Ansätze besser sei als der andere. Beide haben spezifische Vor- und Nachteile. Vor der Entscheidung müssen auch die Anwendungsdomäne und die vorhandenen Ressourcen berücksichtigt werden.

2.1.2 Aufgaben und Aufbau eines IE-Systems

Es gibt eine große Reihe von Aufgeben bei IE, zum Beispiel das Finden eines Firmennamen im Text, oder das Finden einer Krankheit und der dazu gehörenden typischen Symptome, und zu welchem Zeitpunkt diese Symptome erscheinen können. Der IE-Prozess besteht hauptsächlich aus zwei Teilen. Erster ist die Extraktion von „Fakten“ per lokaler Textanalyse aus den Texten. Zweiter ist die Integration von diesen Fakten, wobei neue Fakten entstehen. Am Ende der Integration werden die relevanten Fakten im bestimmten Format umstrukturiert und als Ergebnis ausgegeben.

MUC ist der Name für Message Understanding Conference. Es wurde von der DARPA (The Defense Advanced Research Projects Agency) Ende der 80er Jahre gegründet. In einem sehr realen Sinn hat DARPA teilweise durch die Konzentration auf eine bestimmte Art von Aufgaben dem Gebiet von Informationsextraktion gefunden. Die folgenden fünf Typen von IE-Aufgaben [8] hat es bei MUC-7 definiert.

- Named Entity Recognition (NE) ist die Aufgabe, die Eigennamen (z.B. Personen, Orte, etc.) zu erkennen und klassifizieren. Dabei geht von einem überschaubaren Inventar von Namenskategorien aus, aufgeteilt in Allgemein- und Domänenspezifische Kategorien.
- Coreference Resolution (CO) ist die Aufgabe, Ausdrücke zu identifizieren, die sich auf selbe Entität beziehen. Das heißt, verschiedene Begriffe verweisen auf die selben Objekte. Dabei werden Ausdrücke identifiziert, sowohl

Pronomen als auch Nomen.

- Template Element Construction (TE) ist die Aufgabe, die Informationen, bzw. Elemente über eine Entität zusammen zu fügen.
- Template Relation Construction (TR) ist die Findung von Verbindungen zwischen den Entitäten.
- Scenario Template Production (ST) fügt Entitäten und Relationen zu Fragenstellungs-spezifischen Templates zusammen.

Durch Jahrzehntelange wissenschaftlicher Forschung und Entwicklung wurden die Leistungen bei solchen typischen IE-Aufgaben sehr gut vorangetrieben. Die folgende Tabelle 2.1 zeigt uns die maximalen Ergebnissen [7] bei MUC-6 und MUC-7, wobei F der F-Maß Wert mit gleichem Gewicht von Präzision P und Vollständigkeit R ist.

	NE	**CO**	**TE**	**TR**	**ST**
MUC-6	$F < 97\%$	$R < 63\%$ $P < 72\%$	$F < 80\%$		$F < 57\%$
MUC-7	$F < 94\%$	$F < 62\%$	$F < 87\%$	$F < 76\%$	$F < 51\%$

Tabelle 2.1: Bericht der maximalen Ergebnisse von MUC-6 und MUC-7

Es zeigt uns ein sehr gutes Resultat bei NE-Aufgaben. Aber wegen der großen Formulierungsmöglichkeit unserer natürlichen Sprachen kann man noch keine zufriedenstellenden Leistungen bei anderen Aufgaben mit aktueller Technologie herausbringen.

Obwohl die Aufgaben eines IE-Systems höchst unterschiedlich sind, gibt es Kernelemente. Diese Kernelemente können von fast jedem IE-System benutzt werden, ohne Rücksicht darauf, ob es sich um das Ansatzmodell des Wissen-Ingenieurwesens oder das des maschinellen Lernens handelt.

IE-System hat ein modulares Design. Es ähnelt dem „Pipes-and-filters-Muster“ in der Softwarearchitektur-Technologie. Jedes Modul innerhalb eines IE-Systems funktioniert wie ein Transducer. Es benutzt die Pattern-basierten Regeln, um den einzugebenden Text zu filtern und umzustrukturieren.

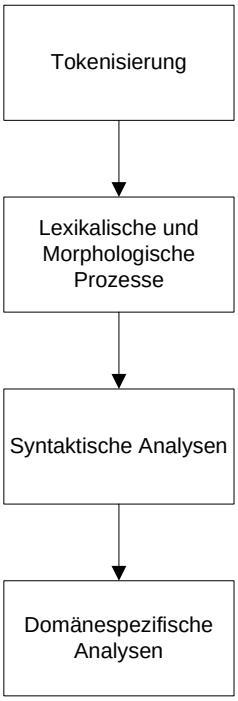

Abbildung 2.1: Allgemeine Struktur eines IE-Systems [3]

Vier primäre Module (siehe Abbildung 2.1) sind in jeden IE-System auffindbar. Die erste Zwei sind ein Tokenizer, einige Arten eines lexikalischen und oder morphologischen Prozesses. Dann folgt die syntaktische Analyse und einige Arten der domänspezifischen Identifizierung. Es gilt zu beachten, dass es einige Systeme gibt, die sich nur nach den ersten zwei Modulen beenden. Es hängt sehr von der genauen Applikation ab.

2.1.3 Evaluationskriterien der Informationsextraktion

Die beiden wichtigen Bewertungskriterien sind die Präzision (Precision) und Vollständigkeit (Recall), diese werden mit folgender Formel definiert.

$$Präzision = \frac{|R \cap P|}{|R|} \tag{2.1}$$

$$Vollstädigkeit = \frac{|R \cap P|}{|P|} \tag{2.2}$$

Sei R sei die Menge relevanter Dokumente, P (für positive)steht für die Menge gefundener Dokumente.

Das "F-Maß"wird mit der Kombination von Präzision p und Vollständigkeit v als Kriterium benutzt.

$$F = \frac{2 \times p \times v}{p + v} \tag{2.3}$$

2.1.4 GATE

Hier stellen wir Ihnen GATE (General Architectur for Text Engineering) kurz vor. GATE ist eine Entwicklungs- und Einsatzinfrastruktur von Softwarekomponenten mit einem Bearbeitungsprozess von menschlicher Sprache. GATE ist ein mit Java entwickeltes Framework. Es wurde im Jahr 1995 von der Universität Sheffield entwickelt und bereits in vielen Forschungs- und Entwicklungsprojekten eingesetzt. GATE wird besonders in vielen Bereichen von Projekten der Informationsextraktion mit unterschiedlichem Sprachen und Domänen verwendet. GATE hat auch eine eingebaute IE-Komponente ANNIE. Deshalb sind viele obergenannte Aufgaben und der Aufbau in GATE abgebildet und leicht für unsere Anwendung direkt zu nutzen.

In GATE sind unterschiedliche Komponenten in den drei Arten von Ressourcen definiert.

- Sprache-Ressource (Language Resources, LRs) repräsentiert die Entitäten wie Lexika, Korpora oder Ontologie;
- Prozess-Ressource (Processing Resources, PRs) repräsentiert die Entitäten. Es sind die Algorithmen, wie grammatikalischer Analysen, Generatoren oder N-Gram Modellierungen;
- Anzeigen-Ressource (Visual Resources, VRs) repräsentiert die Visualisierungs- und Bearbeitungskomponente in der graphischen Benutzeroberfläche.

In GATE sind viele allgemeine Datenstrukturen und Algorithmen als vorhandene Ressource eingebaut. Es sind Dokumente, Korpora und unterschiedliche Annotationstypen, eine Gruppe von Spracheanalysen- Komponenten für Informationsextraktion und eine Reihe von Visualisierungs- und Bearbeitungskomponenten.

GATE unterstützt vielen Datenformate, wie XML, RTF, Email, HTML, SGML und TXT. Alle diese Formate werden analysiert und in einer allgemeinen Model - Annotation - modifiziert. Das Annotationsformat ist eine modifizierte Form des TIPSTER Formates[11]. ANNIE (A Nearly-New Information Extraction) bezeichnet eine Reihe von Prozess-Ressourcen für Sprachanalysen. Mehr Information über ANNIE könne Sie auch In [9] finden.

JAPE (Java Annotation Patterns Engine) bietet „Finite-State-Übertragung“(finite state transduction) über Annotationen, es basiert zudem auf dem regulären Ausdruck. Normalerweise wird dieser an schriftlichen Zeichenketten, einer einfachen Sequenz von Elementen verwendet, aber mit JAPE kann man es für komplexere Datenstrukturen nutzen.

Eine JAPE Regel besteht aus zwei Teilen. Der Linke-Hand-Teil enthält die Annotationsmuster und Operatoren (z.B. *, ?, +) des Regulär Ausdruck [24]. Der Rechte-Hand-Teil besteht aus den Statements, die Annotationen zu manipulieren. Solche Manipulationsstatements kann auch aus Java Code bestehen. Es ist sehr nützlich für die Entfernung von vorübergehenden Annotationen und Feature-Manipulierung der vorhergehenden Annotationen.

Dieses Beispiel 2.1 zeigt uns eine Regel. Wenn eine Annotation als ein Vorname einer Person passt, dann werden die neue Merkmale (Features) wie Geschlecht und Regel, hinzugefügt. Mehr Details werden wir in Kapitel 4 beschreiben.

Listing 2.1: FirstName.jape

```
Rule: FirstName
(
    {Lookup.majorType == person_first}
):person
-->
{
    gate.AnnotationSet person = (gate.AnnotationSet)
       bindings.get("person");
    gate.Annotation personAnn = (gate.Annotation)person.
       iterator().next();
    gate.FeatureMap features = Factory.newFeatureMap();
    features.put("gender", personAnn.getFeatures().get("
       minorType"));
    features.put("rule", "FirstName");
    outputAS.add(person.firstNode(), person.lastNode(),
       "FirstPerson",features);
}
```

Immer mehr NLP (Natur Language Processing) Projekte werden in der Ontologie und taxonomischen Datenstruktur verwendet. GATE verfügt auch über zahlreiche Unterstützung für unterschiedliche Ontologie-Sprachen (z.B. OWL [20], RDF-Schema [18] und DAML-OIL [14]). Bevor wir auf die Ontologie und unser Design und die Implementierung eingehen, werden wir uns einige Grundlagen der Beschreibungslogik ansehen.

2.2 Grundlagen der Beschreibungslogiken

Beschreibungslogiken[4] (Description Logics, DL) ist eine Familien von logikbasierenden Repräsentationssprachen. Dieser aktuellste Name hat die gleichen Be-

zeichnungen wie *terminological logics* und *concept languages*. Beschreibungslogiken werden auf der Basis der Semantik Networks, frame-based Systems und des KL-ONE weiter geleitet. Das Wissen eines Anwendungsgebietes lass sich mit der Hilfe der Beschreibungslogiken leichter und präziser repräsentieren. Sie ist auch Basis von Ontologie-Sprachen wie OWL, die wir in dieser Arbeit verwendet haben. Deshalb machen wir in diesem Kapitel einen kurzen Überblick über die Beschreibungslogiken und der DL-Systems.

2.2.1 Die $\mathcal{AL}$ Sprachfamilie

Eine Beschreibungssprache hat zwei grundlegende Bestandteile, *atomare Konzepte* und *atomare Rollen*. Mit der Hilfe von Konzept-Konstruktoren lassen sich aus diesen atomaren Elementen neue Konzepte erzeugen. So definiert man die Description-Logic-Sprache $\mathcal{AL}$ durch folgende Syntax:

C, D	$\longrightarrow$	A	atomares Konzept
	$\mid$	$\top$	Top-Konzept
	$\mid$	$\bot$	Bottom-Konzept
	$\mid$	$\neg A$	atomare Negation
	$\mid$	$C \sqcap D$	Schnitt
	$\mid$	$\forall R.C$	Wertbeschränkung
	$\mid$	$\exists R.\top$	eingeschränkte existenzielle Quantifizierung

Hierbei ist A eine abstrakte Notation für die atomare Konzepte, und R für die atomare Rollen. C und D bezeichnen die komplexe Konzepte. Zu Beachten ist, dass die Negation $\neg$ nur für atomare Konzepte und der Existenzquantor $\exists$ nur für das universelle Konzept gilt.

Um eine formale Semantik der $\mathcal{AL}$-Konzepte zu definieren, betrachten wir die Interpretation $\mathcal{I}$. Die Interpretation besteht aus einer nicht-leeren Menge $\Delta^{\mathcal{I}}$ (wobei $\Delta^{\mathcal{I}}$ die Domäne der Interpretation ist) und einer *Interpretationsfunktion* $\cdot^{\mathcal{I}}$. Die Interpretationsfunktion gibt für jedes atomare Konzept A eine Menge $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ und zu jeder atomaren Rolle R eine binäre Relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ an. Für Konzept-

beschreibungen gilt folgende induktive Erweiterung der Interpretationsfunktion:

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

Jetzt definieren wir hier die semantische Äquivalenz $X \equiv Y$ und Inklusion $X \sqsubseteq Y$. Wenn $X^{\mathcal{I}} = Y^{\mathcal{I}}$ für alle Interpretationsfunktionen $\mathcal{I}$ gilt, dann sagen wir, X und Y semantisch äquivalent sind, und notieren dies mit $X \equiv Y$. Die Inklusion $X \sqsubseteq Y$ gelte gdw. $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$.Hierbei können X, Y Konzepte oder Rollen sein.

Allein die Aussagekraft von $\mathcal{AL}$ ist beschränkt. Um die Aussagekraft von $\mathcal{AL}$ zu verstärken, hat man weitere Konstruktoren hinzugefügt. Dabei kommen eine Reihe von Beschreibungssprachen heraus.

- Vereinigung $\mathcal{U}$

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \sqcup D^{\mathcal{I}}$$

- volle existenzielle Quantifizierung $\mathcal{E}$

$$(\exists R.C)^{\mathcal{I}} = a \in \Delta^{\mathcal{I}} |\ \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}$$

- Anzahlbeschränkung $\mathcal{N}$

$$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} |\ \sharp\{(a,b) \in R^{\mathcal{I}}\} \geq n\}$$

$$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} |\ \sharp\{(a,b) \in R^{\mathcal{I}}\} \leq n\}$$

- volle Negation $\mathcal{C}$

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

Mit Negation gelten folgende Ausdrücke für die Vereinigung und existenzielle Quantifizierung.

$$\begin{aligned} C \sqcup D &\equiv \neg(\neg C \sqcap \neg D) \\ \exists R.C &\equiv \neg\forall\ R.\neg C \end{aligned}$$

Damit kann man annehmen, dass jede Sprache, die die volle Negation $\mathcal{C}$ enthält auch die Vereinigung und existenzielle Quantifizierung für diese Sprache zu Verfügung stellt. Das heißt, jede $\mathcal{C}$-Sprache ist eine $\mathcal{UE}$-Sprache. Deshalb können wir $\mathcal{ALC}$ statt $\mathcal{ALUE}$, und $\mathcal{ALCN}$ statt $\mathcal{ALEUN}$ schreiben.

2.2.2 Wissensbasen

Eine Wissensbasis hat zwei Komponenten, *TBox* und *ABox*. Die TBox beschreibt die Terminologie einer Domäne. Es ist das Vokabular der Anwendungsdomäne und besteht aus Konzepten und Rollen. Die Rollen beschreiben binäre Beziehungen zwischen Konzepten. Es gibt atomares Konzept und atomare Rolle. Zudem enthält ABox die Individuen bezüglich dieser Terminologie und wird auch Weltbeschreibung genannt.

TBox

Die TBox besteht aus einer Menge von terminologischen Axiomen, welche Aussagen darüber machen, wie Konzepte oder Rollen zueinander in Beziehung stehen. Sie ist auch die Grundlage für die Ontologien. Die speziellen Axiome sind Definitionen, mit denen man Symbolnamen für komplexe Beschreibungen einführen kann. Eine Äquivalenz $A \equiv D$ ist eine Definition gdw. A ein atomares Konzept ist. Eine Menge von Definitionen $\mathcal{T}$ ist genau dann eine *Terminologie* oder *TBox*, wenn es für jedes atomare Konzept A nur ein Axiom in $\mathcal{T}$ gibt, dessen linke Seite A ist.

In einer Terminologie gibt es zwei Arten von Symbole. Namenssymbolen $\mathcal{N}_{\mathcal{T}}$ sind Symbole, die atomare Konzepte auf der linken Seite einer Definition sind. Ba-

sissymbole $\mathcal{B}_\mathcal{T}$ sind die Symbole, die nur auf der rechten Seite einer Definition auftreten.

Mit den Namenssymbolen $\mathcal{N}_\mathcal{T} = \{$ Frau, Mann, Mutter, Vater, Elternteil, Großmutter $\}$ und den Basissymbolen $\mathcal{B}_\mathcal{T} = \{$ Person, Weiblich, Männlich $\}$ können wir folgendes Beispiel der TBox $\mathcal{T}$ definieren.

$$\begin{aligned}
\mathsf{Frau} &\equiv \mathsf{Person} \sqcap \mathsf{Weiblich} \\
\mathsf{Mann} &\equiv \mathsf{Person} \sqcap \mathsf{Männlich} \\
\mathsf{Mutter} &\equiv \mathsf{Frau} \sqcap \exists \mathsf{hatKinder.Person} \\
\mathsf{Vater} &\equiv \mathsf{Mann} \sqcap \exists \mathsf{hatKinder.Person} \\
\mathsf{Elternteil} &\equiv \mathsf{Vater} \sqcup \mathsf{Mutter} \\
\mathsf{Großmutter} &\equiv \mathsf{Frau} \sqcap \exists \mathsf{hatKinder.Eternteil}
\end{aligned}$$

Wegen Konzepte sich nicht vollständig durch andere Konzepte beschreiben lassen, bietet es sich an, die TBox um Inklusion zu erweitern. Die Inklusion, die das atomare Konzept auf die linke Seite hat, nennen wir *Spezialisierung*. Wir bezeichnen solche TBoxen, deren linke Seiten zudem jeweils nur einmal auftreten, als *generalisierte TBox*. Eine einfache Ersetzung erlaubt es uns, eine generalisierte TBox $\mathcal{T}$ in eine sog. normalisierte TBox $\mathcal{T}'$ zu transformieren, indem jede Spezialisierung $A \sqsubseteq D$ durch eine Definition $A \equiv B \sqcap \overline{A}$ ersetzt wird. Dabei bezeichnet $\overline{A}$ ein neues Basiskonzept. Zum Beispiel wird

$$\mathsf{Frau} \sqsubseteq \mathsf{Person}$$

durch die Spezialisierung in eine Definition

$$\mathsf{Frau} \equiv \mathsf{Person} \sqcap \overline{\mathsf{Frau}}$$

umgewandelt.

ABox

Die Zweite Komponente einer Wissensbasis ist die ABox (assertionale Box). Sie besteht aus einer endlichen Menge von Aussagen über Individuen, die extensionales

Wissen über die Welt bezüglich einer TBox beschreibt. ABox enthält zwei Arten von Axiomen. Eine ist die sog. *Instanz-* oder *Konzept-Assertion*, und die Andere *Rolle-Assertion*. Hier kennzeichnen wie die Individuen mit a, b, c, das Konzept mit C und die Rolle mit R. Dann können wir mit folgenden Formen die Aussagen in einer ABox bezeichnen.

- $C(a)$, die Instanz- oder Konzept-Assertion
- $R(b, c)$ die Rolle-Assertion

Wenn zum Beispiel JOHN, SARAH die Namen von Individuen sind, dann können wir eine ABox wie folgende definieren.

$$\mathsf{Weiblich(SARAH)} \qquad \mathsf{Vater(JOHN)} \qquad \mathsf{hatKind(JOHN,\ SARAH)}$$

Somit erlaubt die ABox die Formulierung einer konkreten Situation bezüglich des terminologischen Rahmens der TBox und wird deshalb auch als *Weltbeschreibung* bezeichnet.

Die Bedeutung der Konzept-Assertion muss auch in der Semantik berücksichtigt werden. Dafür erweitern wir die Definition um eine Interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ indem wir zu jedem Individuensymbol a seine Interpretation $a^{\mathcal{I}}$ mit in $\Delta^{\mathcal{I}}$ aufnehmen. Um Mehrdeutigkeit zu vermeiden muss dabei die Einschränkung von $\cdot^{\mathcal{I}}$ auf die Individuensymbole injektiv sein. Eine Konzept-Assertion $C(a)$ ist *erfüllt* gdw. $a^{\mathcal{I}} \in C^{\mathcal{I}}$ gilt. Und Ein Rolle-Assertion $R(b, c)$ ist *erfüllt* gdw. $(b, c) \in R^{\mathcal{I}}$ gilt. Des weiteren erfüllt $\mathcal{I}$ die gesamte ABox $\mathcal{A}$ genau dann, wenn $\mathcal{I}$ jede einzelne Aussage und Rolle von A erfüllt. Man spricht dann auch von $\mathcal{I}$ als Modell von $\mathcal{A}$.

2.2.3 Standard-Inferenzdienste

Eine Eigenschaft der Beschreibungslogiken ist eine wohl definierte Semantik, die auf Logik basiert. Diese entspricht einem entscheidbaren Fragment der Logik erster Stufe. Eine wichtige Eigenschaft, die die Beschreibungslogiken außerdem auszeichnet ist die Unterstützung von Inferenzdienste, die das automatische Reasoning ermöglichen. Bevor wir über die Standard-Inferenzdienste eines DL-Systems

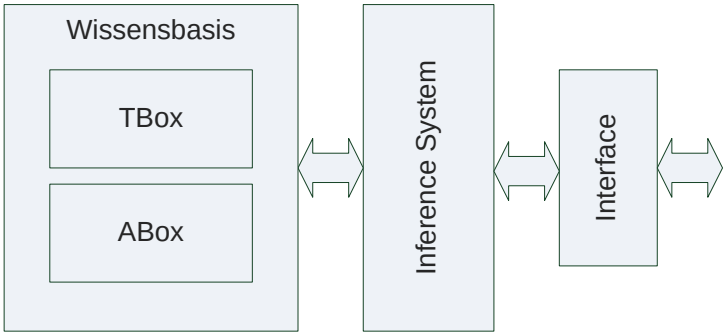

Abbildung 2.2: Architektur eines Standard DL-Systems [4]

sprechen, werfen wir zuerst einen Blick auf die den Architektur eines Standard DL-Systems.

Ein Standard DL-System besteht aus einer Wissensbasis, einem Inferenzsystem und einem Interface (Siehe Abbildung 2.2). Wie wir bereits kennengelernt haben, beinhaltet die Wissensbasis zwei Komponenten: die TBox und die ABox. Die Inferenzdienste eines DL-Systems ist gleichzeitig für die TBoxen und die ABoxen verfügbar.

Als wichtigsten Dienst ist der *Erfüllbarkeitstest* bzw. *Konsistenztest* für Konzeptterme. Die Inkonsistenz bestimmter Konzeptnamen deutet auf Modellierungsfehler bei der Erstellung eines terminologischen Modells hin. Ein weiterer Dienst ermittelt die *Subsumierungsbeziehung* zwischen Konzepttermen. Die Subsumierung von Konzepttermen ist für den Aufbau einer sog. Taxonomie (sog. Subsumierungshierarchie) bedeutsam. Für viele Anwendungen ist es bedeutsam, nicht nur Aussagen über Konzepte zu machen, sondern Aussagen über Individuen zu verwalten. Der wichtigste Dienst ist hier die Überprüfung der ABox auf Konsistenz. Ein weiterer Dienst ist das Finden aller Individuen, die Instanz von einem gegebenen Konzept sind. Die folgende Liste zeigt die genauen Definitionen von einigen solcher Standard-Inferenzdiensten [4].

Erfüllbarkeit: Ein Konzept C heißt erfüllbar bzgl. einer TBox $\mathcal{T}$ gdw. ein Modell $\mathcal{I}$ von $\mathcal{T}$ existiert, so dass $C^{\mathcal{I}} \neq \emptyset$.

Subsumierung: Ein Konzept D subsumiert ein Konzept C bzgl. einer TBox $\mathcal{T}$ gdw. für jedes Modell $\mathcal{I}$ von $\mathcal{T}$ die Inklusion $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ gilt. Hier schreiben wir $C \sqsubseteq_{\mathcal{T}} D$ oder $\mathcal{T} \models C \sqsubseteq D$.

Äquivalenz: Zwei Konzepte C und D sind äquivalent bezüglich einer TBox $\mathcal{T}$ gdw. für jedes Modell $\mathcal{I}$ von $\mathcal{T}$ die Gleichheit $C^{\mathcal{I}} = D^{\mathcal{I}}$ gilt. Wir schreiben dafür $C \equiv_{\mathcal{T}} C$ oder $\mathcal{T} \models C \equiv D$.

Disjunktheit Die Konzepte C, D sind disjunkt bzgl. einer TBox $\mathcal{T}$ gdw. für jedes Modell $\mathcal{I}$ von $\mathcal{T}$ gilt: $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$

Für die praktische Umsetzung der Inferenzdienste gibt es weitere Informationen im Buch „The Description Logic Handbook“ [4].

2.2.4 Closed-World- und Open-World-Semantik

Man kann die Wissensbasis eines DL-Systems mit Datenbanken vergleichen, weil die Art und Weise, um Wissen zu repräsentieren bei beiden sehr ähnlich scheint. Das Schema einer Datenbank ähnelt der TBox. Die expliziten Datensätze der Datenbank kann man mit der ABox vergleichen. Der große Unterschied liegt jedoch in der verwendeten Semantik zur Wissensrepräsentation. Datenbanken verwenden eine *Closed-World-Semantik*. Das heißt, eine Instanz in der Datenbank repräsentiert exakt eine Interpretation, weil die Entitäten und Relationen dort in eindeutiger Weise als Mengen und Relationen interpretiert werden. Im Gegensatz wird eine *Open-World-Semantik* bei den ABoxen verwendet. Die Aussagen einer ABox bestimmen die Eigenschaften, die viele verschiedene Interpretationen gemein haben müssen, um ein Modell von Allem zu Sein. Wenn eine Information nicht in der Datenbank ist, dann gilt sie als falsch. Aber die Fehlende Information in einer ABox gilt nur las unvollständiges Wissen.

Wenn zum Beispiel hatKinder(PETER, HARRY) die einzige Assertion über Peter in einer Datenbank ist, dann bedeutet das in der eindeutigen Interpretation, dass Peter genau ein Kind hat. In einer ABox sagt diese Assertion uns nur, dass Harry ein Kind von Peter ist. Der ABox hat andere Modelle in denen Harry das einzige Kind von Peter ist, es gibt aber, in denen Harry viele Geschwister hat. Die Tatsache, dass

Peter nur ein Kind hat, müsste durch die explizite Aussage $(\leq 1\mathsf{hatKinder})(\mathsf{Peter})$ ausgedrückt werden.

Das Open-World-Reasoning benötigt die Fallunterscheidungen, die in Closed-World-Reasoning unnötig sind. Deshalb sind die Inferenzaufgaben in DL-Systemen im Allgemeinen komplexer als die Anfrageverarbeitung in Datenbanksystemen.

Kapitel 3

Analyse und Entwurf

Als eine wichtige Informationsquelle für weitere Design- und Implementierungsphasen ist das Anwendungsszenario der Startpunkt unseres Projekts. Dieses Kapitel beschreibt zunächst dieses ausgewählte Szenario. Zu diesem Szenario werden vier klinische Studien für die weitere Phase des Entwicklungsprozesses ausgewählt und im Abschnitt 3.2 analysiert. Dann stellen wir Ihnen die Entwürfe der Systemarchitektur und Wissensbasis vor. Im Anschluss präsentieren wir Ihnen das Design der Benutzeroberfläche.

3.1 Anwendungsszenario

Nehmen wir an, im Intranet eines Krankenhaus oder eines Pharmaunternehmens gibt es eine Dokumentablage von klinischen Studien. Bei jeder dieser Studien handelt es sich um eine Untersuchung mit zwei Medikamenten oder zwei Kombinationen von Medikamenten. Um den Zugang zu diesen Studien für die Mitarbeiter zu erleichtern, extrahiert eine Ontologie-basierte Softwarekomponente die Informationen aus diesen klinischen Studien, so dass dabei eine Wissensbasis entsteht. Als ein Beispiel dient folgender Satz einer klinischen Studie:

> Mean blood pressure was lower in the telmisartan group than in the

> placebo group throughout the study (weighted mean difference between group 4.0/2.2 [SD 19.6/12.0]mm Hg). [2]

Für diese klinische Untersuchung gibt es zwei Patientengruppen. Dabei ist eine Gruppe mit dem Medikament Telmisartan behandelt worden und die andere mit einem Placebo. Als Resultat der Studie zeigt dieser Satz, dass der durchschnittliche Blutdruck der Patienten der Telmisartan-Gruppe niedriger ist als der der Placebo-Gruppe. In diesem Satz ist die einfache Kenntnis enthalten, dass es sich bei dieser klinischen Studie um einen Vergleich mit Telmisartan und Placebo handelt. Genau diese Kenntnis könnte als Referenz für andere Artikel gebraucht werden. Deshalb wollen wir durch die Informationsextraktion dieses Wissen aus dem Text extrahieren, um dann das explizite Wissen in eine Wissensbasis für die Anwender bereit zu stellen.

Ein Arzt oder Mitarbeiter startet ein Textverarbeitungsprogramm und schreibt einen Artikel über die Wirkung eines Arzneimittels, zum Beispiel Telmisartan. Dabei wird er eine klinische Studie aus dem Korpus als Referenz angeben. In diesem Textverarbeitungsprogramm ist eine Komponente als Zugang zu dieser Wissensbasis bereitgestellt. Mit einem Wort „Telmisartan“ fragt der Benutzer alle Studien, die einen Vergleich von Telmisartan mit einem anderen Medikament beinhalten, in der Wissensbasis ab. Dabei bekommt er eine Titelliste als Suchergebnis zurück. Er wählt ein Dokument in dieser Liste aus. Wenn diese Studie passend für ihn ist, wird sie mit einem Mausklick automatisch als Referenz in seinem Artikel angegeben. An der gewünschten Stelle wird eine Zitation eingetragen, gleichzeitig wird dem Literaturverzeichnis ein Eintrag mit Titel, Autorenname und anderen Meta-Informationen der ausgewählten klinischen Studie zugeordnet.

3.2 Testdokumente

Für die Implementierung werden vier klinische Studien in englischer Sprache als Test-Dokumente verwendet. Die komplexe Testdokumente finden Sie in Anhang B. Jede dieser Studien behandelt einem Vergleich von zwei Medikamenten oder zwei Kombinationen aus mehreren Medikamenten. Für das Experiment dieser Arbeit sind diese Sätze die Hauptträger der Information. Natürlich gibt es auch sehr viele

weitere Informationen in der gleichen klinischen Prüfung, die von unterschiedlichen Lesern als wichtig betrachten werden. Aus zeitlichen und inhaltlichen Gründen können wir keine volle Untersuchung über diese Studien durchführen. Deshalb wird nur ein Abschnitt vom Resultat aus jeder dieser vier klinischen Studien B in dieser Arbeit verwendet. Jeder der folgenden vier Sätze ist das Resultat aus diesen vier Abschnitten und enthält die wichtigsten Informationen für diese Arbeit. Die blaue Markierung deutet auf die Medikamentennamen hin und die rote Markierung auf die Vergleichsrelationen.

> Mean blood pressure was lower in both the telmisartan group (a 0.9/0.6 mm Hg greater reduction) and the combination-therapy group (a 2.4/1.4 mm Hg greater reduction) than in the ramipril group. [15]
>
> Mean blood pressure was lower in the telmisartan group than in the placebo group throughout the study (weighted mean difference between group 4.0/2.2 [SD 19.6/12.0]mm Hg). [2]
>
> Long-term administration of clopidogrel to patients with atherosclerotic vascular disease is more effective than aspirin in reducing the combined risk of ischaemic stroke, myocardial infarction, or vascular death. [1]
>
> Overall, clopidogrel plus aspirin was not significantly more effective than aspirin alone in reducing the rate of myocardial infarction, stroke, or death from cardiovascular causes. [6]

Die Medikamente, die hier behandelt werden sind telmisartan, ramipril, clopidorgrel und aspirin. Ein placebo [22] im engeren Sinne ist eine Tablette oder anderes medizinisches Präparat, welches keinen pharmazeutischen Wirkstoff enthält und somit per Definition auch nicht durch einen solchen Stoff eine pharmazeutische Wirkung verursachen kann. Es wird hier eingesetzt, um die pharmazeutische Wirksamkeit von Medikamenten genau zu untersuchen. Sämtliche Textfragmente sollen als Entitäten erkannt werden. Im vierten Satz wird „clopidogrel plus aspirin“ nicht als zwei Einheiten von Medikamenten für den Vergleich betrachtet, sondern als eine Kombination aus zwei Medikamenten.

Aus obigen Sätzen wollen wir im späteren Experiment eine Information gewinnen, nämlich zwischen welchen Medikamenten es einen Vergleich in welcher klinischen

Studie gibt.

3.3 Systemarchitektur

Das Anwendungsszenario zeigt deutlich drei Anforderungen des Systems. Als Erste Anforderung sollen die Informationen aus den Texte extrahiert werden, und zwar nicht nur die benannten Entitäten, sondern auch die Relation zwischen diesen. Im ersten Satz beispielsweise sollen bei der Informationsextraktion die Wörter „telmisartan“ und „ramipril“ als Medikamente identifiziert (Siehe Abbildung 3.1. Gleichzeitig muss durch das Wort „than“ auch die Vergleichsrelation erkannt werden. Die extrahierten Informationen werden dann in einer Wissensbasis gespeichert.

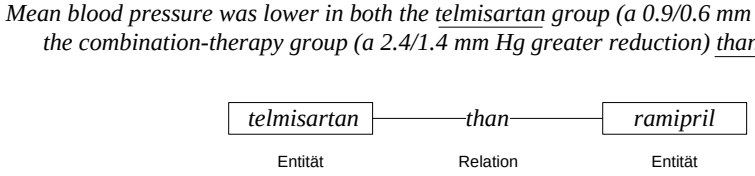

Abbildung 3.1: Informationsextraktion von Entitäten und Relation aus einem Beispielsatz

Deshalb wird diese Aufgabe in drei Phasen verteilt. Die erste Phase ist die Informationsextraktion (Siehe Abbildung 3.2). Eine selbst definierte GATE Applikation verwendet eine Ontologie, um die Entitäten und Relationen in den Textdokumenten (klinische Studien) zu annotieren. Die Annotationen werden an einen ABox Generator weitergeleitet. Dieser ABox Generator ist ein selbst entwickeltes Java Programm mit Ontologie API von Jena Framework. Der ABox Generator übernimmt diese Annotationen und generiert für jedes Dokument eine Analyze-ABox. In dieser ABox werden die gesuchten Entitäten und Relationen als einfache Individuen repräsentiert.

Die Analyze-ABoxen werden dann mit passenden Regeln von einer Inferenzmaschine namens RACER interpretiert. Solche Regeln nennen wir *Interpretationsregeln*. Am Ende dieser Interpretationsphase (Siehe Abbildung 3.3) entsteht zu jedem Dokument eine neue interpretierte ABox. Jetzt sind diese interpretierten ABoxen

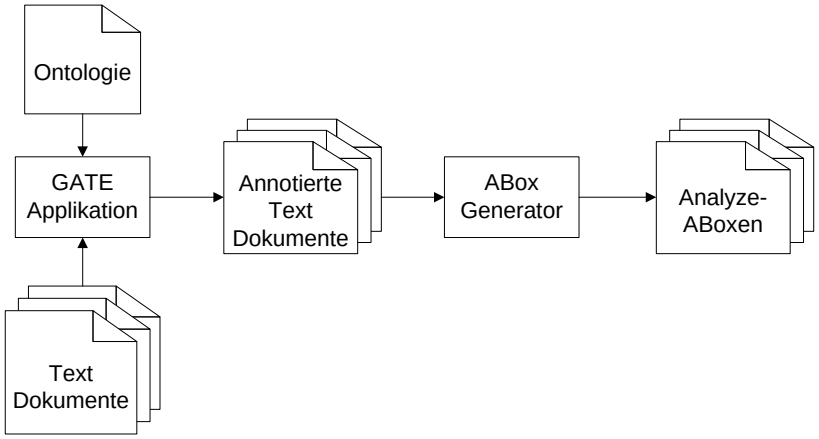

Abbildung 3.2: Informationsextraktionsphase

für die weitere *Materialisierungsphase* (Siehe Abbildung 3.4) bereit. In der dritten Phase werden die interpretierten ABoxen zusammen mit TBox (Ontologie) von RACER in einen Triplestore umgewandelt.

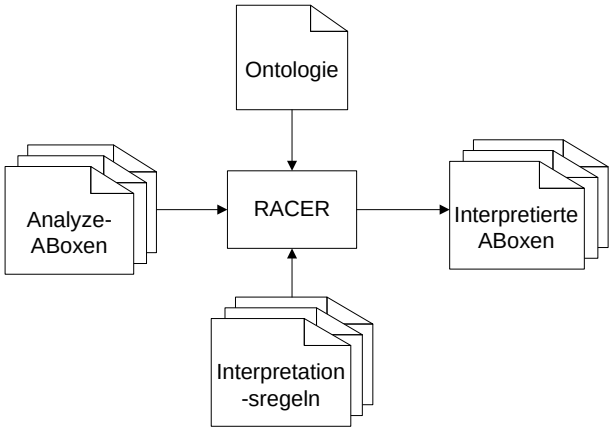

Abbildung 3.3: Interpretationsphase

Die zweite Anforderung ist eine Schnittstelle für den Anwender. Dadurch kann er Dokumente suchen und die passenden Inhalte direkt als Referenz im eigenen Dokument angeben. Eine integrierte Softwarekomponente im Textverarbeitungsprogramm bietet dem Benutzer einen Zugang zu der Wissensbasis. Hier haben wir ein Add-In für Microsoft Word 2007 als Client Programm entwickelt. Wie bei einer

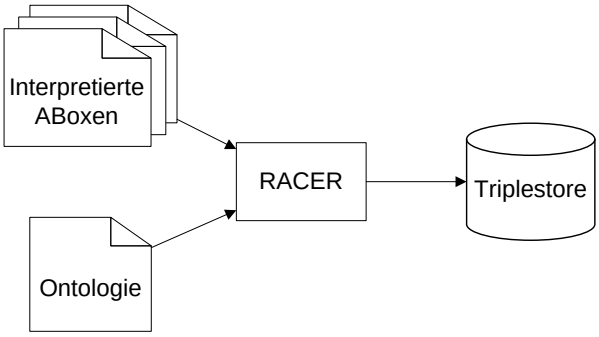

Abbildung 3.4: Mateiraliserungsphase

normalen Suchmaschine gibt der Anwender einen gewünschten Suchbegriff ein und startet die Suche.

Per HTTP und SOAP wird der Suchbegriff an einen Applikations-Server gesendet. Der Suchprozess ist ein Webservice auf diesem Server. Eine passende Abfrage wird für den Suchbegriff generiert und weiter an den AllegroGraph RDFStore Server geleitet. Der AllegroGraph RDFStore Server arbeitet wie ein herkömmlicher Datenbank-Server und liefert die passenden Daten als Antwort zurück. Für den Referenzzweck sind die Antworten die Meta-Informationen von Dokumenten. Der Applikations-Server übernimmt diese Meta-Informationen und liefert diese in einem passenden XML Format an den Client zurück. Der Anwender kann dann je nach Inhalt ein bestimmtes Dokuments als Referenz in der eigenen Arbeit angeben.

Alle Komponenten und Prozesse werden zusammengesetzt. Abbildung 3.5 zeigt einen Überblick der gesamten Systemarchitektur.

3.4 Design der Benutzeroberfläche

Die Benutzeroberfläche (*Graphical User Interface, GUI*) ist das Gesicht eines Softwareprodukts. Die möglichen Bedienungsaktionen des Benutzer spielen beim Design der Benutzeroberfläche eine wichtige Rolle. Ein Anwendungsfalldiagramm (Siehe Abbildung 3.6) erfasst die möglichen Aktionen beim Suchen und Zitieren in Rahmen des vorgelegten Szenarios.

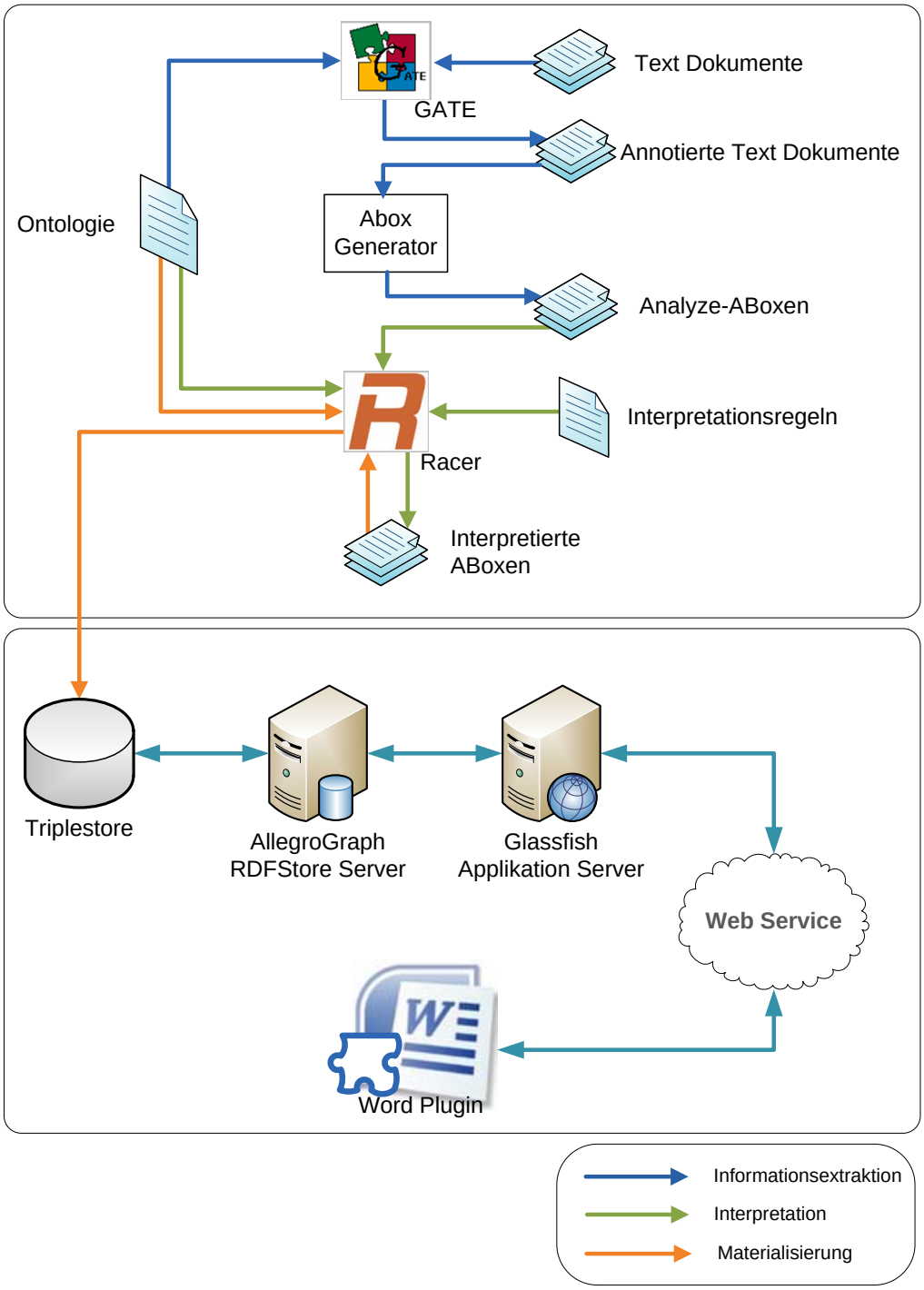

Abbildung 3.5: Überblick der Systemarchitektur

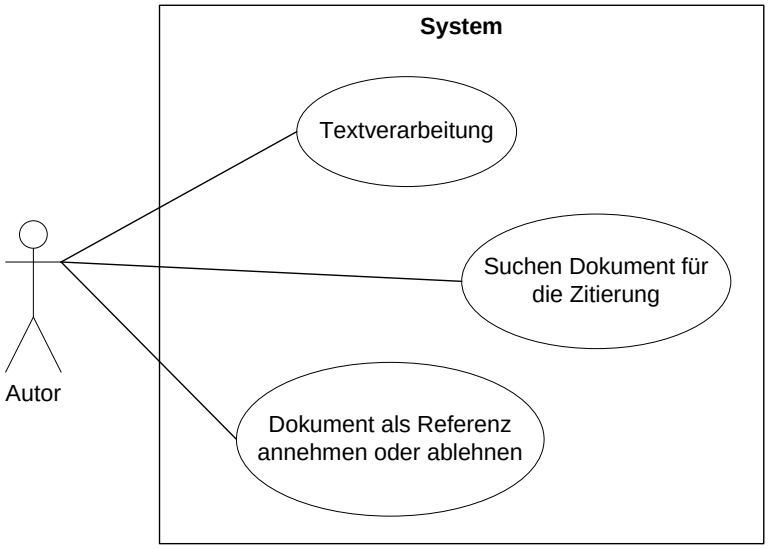

Abbildung 3.6: Anwendungsfalldiagramm der Intergrationskomponente in Word

Eine integrierte Komponente in Microsoft Word 2007 [23] ist die Benutzeroberfläche dieser Arbeit. Als weit verbreitetes Textverarbeitungsprogramm ist Microsoft Word den meisten Computeranwendern bekannt. Die gute Erweiterbarkeit ermöglicht, die selbst definierten Komponenten mit neuer Funktionalität und einheitlichem Stil der Benutzeroberfläche in Word zu integrieren.

Ein ActionsPane ist der Hauptteil unseres semantischen Referenztools für den Anwender (Siehe Abbildung 3.7). Hier kann er den gewünschten Suchbegriff in einer Form von Freitext in das Textfeld eingeben und als Anfrage an den Server senden. Die Ergebnisse zeigen sich in einer Liste dieses ActionsPanes. Ein weiterer Preview-Bereich zeigt den Inhalt eines ausgewählten Dokuments an. Nach der Überprüfung des Inhalts kann dieses Dokument per Mausklick direkt und ohne Kenntnis der Meta-Informationen als Referenz in der eigenen Arbeit angegeben werden.

Die neue ActionsPane wird dem Anwender nur bei dem Bedarf angezeigt. Eine Erweiterung auf den Multifunktionsleisten (im Original „Ribbons“)[23] dient als Schalter der neuen Funktionalität.

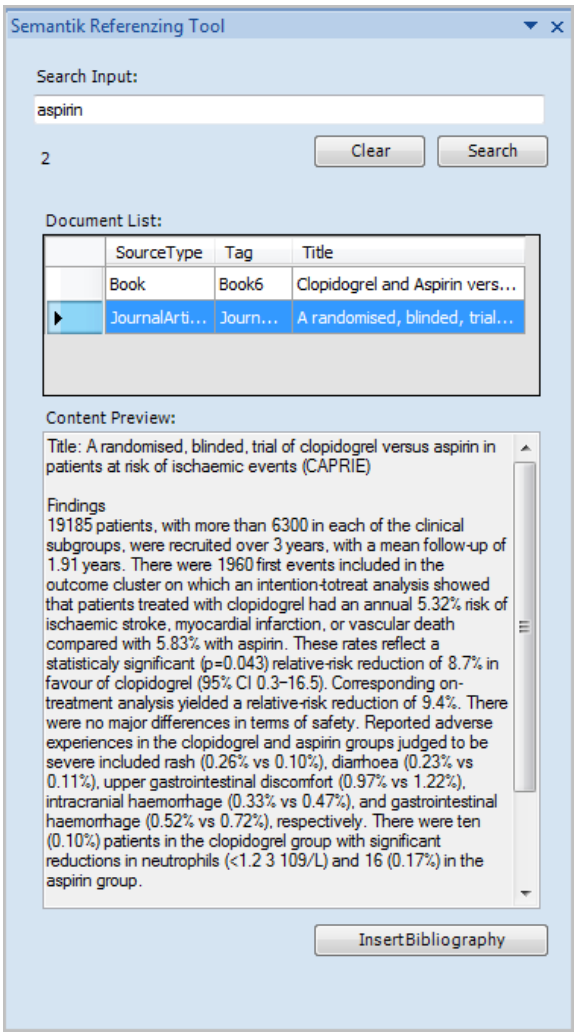

Abbildung 3.7: ActionPane vom semantischen Referenztool in Word 2007

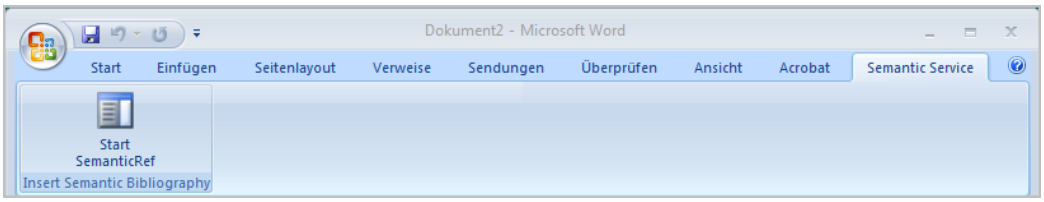

Abbildung 3.8: Multifunktionsleisten in Word 2007

Kapitel 4

Implementierung

Nach Analyse und Design beginnt die Implementierung des Projekts mit der Modellierung einer Ontologie. Die wichtigen Prozessphasen des Systems haben wir bereits in Kapitel vorgestellt. Im folgenden Kapitel werden diese Prozessphasen in der Implementierung des Systems detailiert beschrieben.

4.1 Modellierung der Wissensbasis

Nach der Analyse der vier klinischen Studien und der Anforderung des Projekts definieren wir eine TBox mit der folgenden Struktur (Siehe Abbildung 4.1). Diese TBox liegt in Form einer Ontologie vor und ist die Zentrale der gesamten Wissensbasis.

Die TBox wird in dieser Arbeit mit Hilfe von Protégé [17] in OWL erstellt. Das Akronym OWL steht für *Web Ontology Language* [21]. OWL ist ein XML basierter Sprach-Standard des W3C zu Erstellung, Publizierung und Verteilung von axiomatisierten formalen Ontologien. Bei der Schaffung von OWL spielt das Gleichgewicht zwischen Ausdrucksstärke der Sprache einerseits und effizientem Schlussfolgern - d.h. Skalierbarkeit - andererseits eine tragende Rolle. Mit drei unterschiedlichen Ausdrucksstärken wird OWL in die Teilsprachen *OWL List*, *OWL DL* (description logic) und *OWL Full* geteilt. OWL Lite ist zur Repräsentation einfacher Taxono-

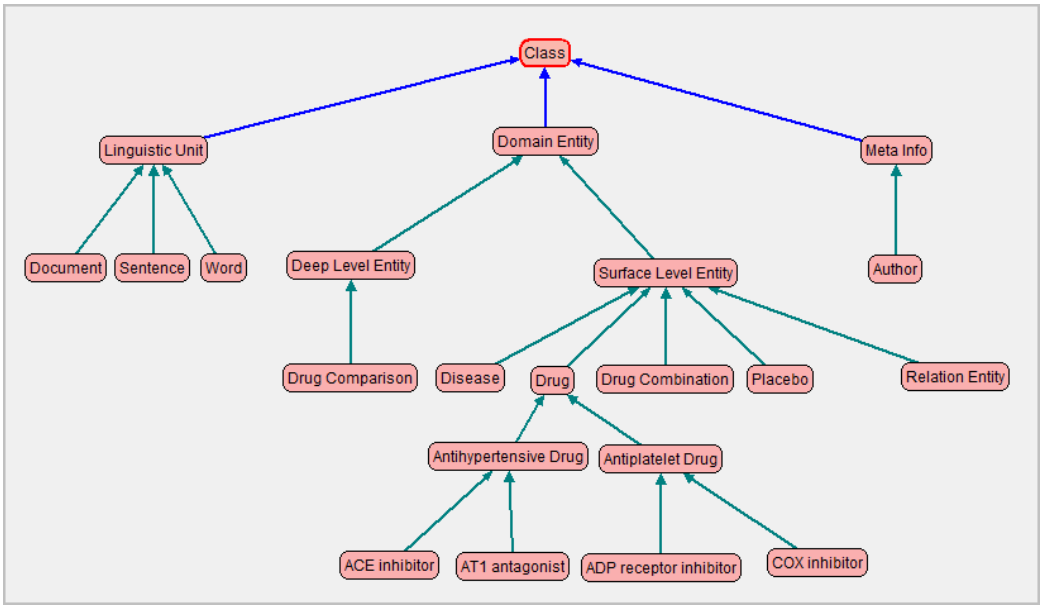

Abbildung 4.1: Klassenhierarchie der TBox

mien (Ober-/Unterbegriffshierarchien) gedacht. OWL DL hat die Mächtigkeit von terminologischen Beschreibungslogiken. Mit der Einschränkung auf eine Untermenge der Prädikatenlogik 1.Stufe lassen sich Konsistenz und Schlussfolgerungen in endlicher Zeit automatisch überprüfen (Entscheidbarkeit). Wie der Name bereits sagt, ist OWL Full Web Ontology Language ohne weitere Einschränkung. Die wichtigsten Vor- und Nachteile der drei Teilsprachen von OWL sind in folgender Liste [13] dargestellt.

OWL Full:
- enthält OWL DL und OWL Lite,
- enthält als einzige OWL-Teilsprache ganz RDFS,
- sehr ausdrucksstark,
- Semantik enthält einige Aspekte, die aus logischem Blickwinkel problematisch sind,
- unentscheidbar,
- wird durch aktuelle Softwarewerkzeuge nur bedingt unterstützt.

OWL DL:
- enthält OWL Lite und ist Teilsprache von OWL Full,

- entscheidbar,
- wird von aktuellen Softwarewerkzeugen fast vollständig unterstützt,
- Komplexität NExpTime (worst-case).

OWL Lite:
- ist Teilsprache von OWL DL und OWL Full,
- entscheidbar,
- weniger ausdrucksstark,
- Komplexität ExpTime (worst-case).

Die Grundbausteine von OWL bestehen aus Klassen, Properties (auch Rollen genannt) und Individuen. Klassen werden in OWL durch `owl:Class` [25] definiert. Beispielsweise kann für die Author-Klasse folgende Kurzform verwendet werden.

```
<owl:Class rdf:about="Author" />
```

Rollen in OWL gibt es in zwei Arten: abstrakte und konkrete [5]. Abstrakte Rollen verbinden Individuen mit Individuen, Konkrete Rollen hingegen verbinden Individuen mit Datenwerten. Die Bereiche, auf die sich Rollen beziehen, können mittels `rdfs:domain` und `rdfs:rangs` explizit deklariert werden, wie das folgende Beispiel zeigt.

```
<owl:ObjectProperty rdf:about="hasAuthor">
  <rdfs:domain rdf:resource="Document" />
  <rdfs:range rdfs:resource="Author" />
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="authorFirstName">
   <rdfs:domain rdf:resource="Author"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

Für diese Arbeit werden drei Basis-Klassen, DomainEntity, LinguisticUnit und MetaInfo für unterschiedliche Zwecke in der Ontologie definiert. Die erste Klasse DomainEntity beinhaltet die domänspezifischen Instanzen, in unserem Beispiel Instanzen aus dem medizinischen Bereich. Die Klasse LinguisticUnit entspricht den sprachlichen Elementen des Dokuments. Für die Referenzangabe definieren wir eine dritte Klasse MetaInfo.

DomainEntity hat zwei Unterklassen, SurfaceLevelEntity und DeepLevelEntity. Diese Klassenhierarchie wird in der Beschreibungslogik mit *inclusion* in folgenden Formen dargestellt.

$$\mathsf{SurfaceLevelEntity} \sqsubseteq \mathsf{DomainEntity}$$

$$\mathsf{DeepLevelEntity} \sqsubseteq \mathsf{DomainEntity}$$

Die direkt extrahierbaren Instanzen aus den Texten, beispielsweise die Medikamenten- und Krankheitsnamen, werden als die Unterklassen von SurfaceLevelEntity definiert. Wenn das Wort „than“ sich zwischen zwei Medikamenten befindet, zeigt dies eine Vergleichsrelation zweier Medikamente. Davon benötigen wir die richtige Interpretation eines Satzes. Deshalb wird das Wort „than“ bei der Extraktionsphase nur als Instanz von RelationEntity identifiziert. Der folgende Satz aus dem Testdokukment enthält das Wort „than“, dieses hat hier jedoch nicht die Bedeutung eines Vergleiches von Medikamenten.

> 19185 patients, with more than 6300 in each of the clinical subgroups, were recruited over 3 years, with a mean follow-up of 1.91 years.

Deshalb werden diesen Relationen später in einer Inferenzmaschine mit Regeln genau interpretiert. Damit wollen wir die falsche Interpretation von Relationen vermeiden. In der Interpretationsphase wird die Vergleichsrelation von zwei Medikamenten dann bei passender Regel als neue Instanz von DrugComparison interpretiert. Weil diese Information eine Ebene tiefer liegt als direkt extrahierbare Instanzen wird DeepLevelEntity als Oberklasse definiert.

Die LinguisticUnit-Klasse hat drei Unterklassen: Word, Sentence und Document. Ein Dokument kann einem Satz oder mehrere Sätze enthalten, sowie ein Satz ein Wort oder mehrere Wörter. Dafür wird die Rolle contains in der TBox definiert. Zwischen einer Word-Instanz und einer SurfaceLevelEntity-Instanz existiert immer eine depictsSLE Rolle (`owl:ObjectProperty` in OWL).

Für die Zitation und Literaturliste wird Titel, Name des Autors, des Verlags und zahlreiche anderen Meta-Informationen über ein Dokument benötigt. Deshalb wird MetaInfo als Basisklasse definiert. Die Kinderklasse von MetaInfo ist

Author. Jedes Dokument wird von einem oder mehreren Autoren geschrieben. Die Property hasAuthor entspricht genau dieser Beziehung zwischen Document und Author. Alle anderen Meta-Informationen werden als konkrete Rollen (so genannte `DatatypeProperty`)in TBox definiert.

Nach der Erstellung von TBox, ist jetzt die Informationsextraktion als nächster wichtiger Schritt der Implementierung geplant.

4.2 Informationsextraktion

Als Informationsextraktion wird eine *Corpus Pipeline Application* in GATE definiert. Sie besteht aus fünf Prozess-Ressourcen, nämlich *English Tokeniser*, *Sentence Splitter*, *OntoGazetteer* und zwei *JAPE Transducer*. Als Language Ressourcen dienen die vier ausgewählten Texte und TBox.

Der English Tokeniser und Sentence Splitter sind vorhandene Prozess-Ressourcen in ANNIE von GATE [9]. Der Tokeniser zerlegt den Text in elementare Token, wie Zahlen, Interpunktion, Wörter verschiedenen Typs und Leerzeichen. In GATE werden folgende Typen von Token bereits definiert.

Word: wird als eine Menge von zusammengeschriebenen Buchstaben definiert. Innerhalb eines Worts ist der Bindestrich das einzige erlaubte Sonderzeichen. Das Attribut „`orth`“ deutet auf eine weitere Typinformation einer Word Annotation hin:

- `upperInitial`: erster Buchstabe in Großeschrift, die restlichen in Kleinschrift
- `allCap`: alle Buchstaben in Großschrift
- `lowerCase`: alle Buchstaben in Kleinschrift
- `mixedCaps`: sonstige Kombination von Buchstaben

Number: wird definiert als alle Kombinationen aufeinanderfolgender Ziffern.

Symbol: Eine beliebige Anzahl aufeinanderfolgender Symbole(z. B. & $). Mit dem Attribut „`symbolkind = currency`“ werden nur Währungssymbole wie $ und £ extra gekennzeichnet.

Punctuation: Es werden drei Typen davon definiert, öffnende Satzzeichen(z.B. „{“), schließende Satzzeichen (wie „}“) und normale Satzzeichen. Das Attribut „`position`“ kennzeichnet das öffnende Satzzeichen mit dem Wert "`startpunct`", und das schließende mit „`endpunct`“.

SpaceToken: Leerräume (white spaces) werden ebenfalls in zwei Untertypen, Leerzeichen und Kontrollzeichen, genau klassifiziert. Ein Tabulatorzeichen und ein Zeilenumbruch werden mit „*kind =control*“ als Kontrollzeichen gekennzeichnet, ebenfalls wie ein Leerzeichen mit „*kind=space*“.

Alle klinischen Studien, die wir verwendet haben, sind in englischer Sprache verfasst. Deshalb verwenden wir hier einem English Tokeniser. Mit einem normalen Tokeniser und einem JAPE Transducer ist der English Tokeniser eine verbesserte Lösung für die Texte auf Englisch. Der English Tokeniser erkennt die einzelnen Buchstaben, wogegen der SentenceSplitter die Sätze in einem Text erkennen kann. Dabei wird jeder erkannte Satz im Typus von „Sentence“ annotiert.

Der OntoGazetteer ist eine Prozess-Ressource [16]. In diesem Prozess können die Instanzenlisten der Ontologiekonzepte geladen werden. Für die Funktionalität des OntoGazetteer benötigen wir eine oder mehrere Gazetteer-Listen (.lst Dateien) und zwei .def Dateien. Die beiden Dateien sind eine Mapping-Datei mit dem Namen „mappings.def“ und eine Indexdatei mit dem Namen „lists.def“.

Eine Gazetteer-Liste ist eine einfache Textdatei mit nur einem Eintrag pro Zeile für Firmen-, Personen-, Ortsnamen und zahlreiche weitere spezifische Namen. Jede Liste repräsentiert eine Menge von bestimmten Klassen, deshalb kann man eine Gazetteer-Liste als eine Instanzenliste betrachten. Das folgende Beispiel ist die Gazetteer-Liste für Aspirin als eine Instanz für die Klasse cyclooxygenase.

Listing 4.1: Beispiel Gazetteer-Liste cyclooxygenase.lst

```
1 Aspirin
2 aspirin
3 ASA
```

```
4 acetylsalicylic acid
5 Acetylsalicylic acid
```

Die `mappings.def`-Datei beschreibt die Verbindung zwischen der Gazetteer-Liste und dem Ontologiekonzept. Jeder Eintrag in `mappings.def` hat eine Form mit drei Bestandteilen, die durch „:“ getrennt sind. Der erste Bestandteil ist der Dateiname einer Gazetteer-Liste. Der Zweite ist der Name einer Ontologiedatei.Dritter Bestandteil ist ein Konzeptname in dieser Ontologie. Im Anhang dieser Arbeit finden Sie die mappings.def und lists.def, die wir in dieser Arbeit verwendet haben.

`.lst Datei:Ontologie Datei:Konzeptname`

Sieben Gazetteer-Listen werden in unserer Anwendung definiert. Eine Indexdatei (lists.def) wird verwendet, um auf dieser Listen zuzugreifen. Diese Indexdatei bestimmt die Relationen zwischen der Gazetteer-Liste und dem Annotation-Feature, die von dem OntoGazetteer generiert wird. Wie in der Mapping-Datei hat jeder Eintrag hier auch eine ähnliche Form aus zwei oder drei Bestandteilen.

`.lst Datei:Feature1[:Feature2]`

Bei der Ausführung von OntoGazetteer-Prozess wird eine Annotation für jede Instanz, die sich gleichzeitig in dieser Gazetteer-Liste und in einem Dokument befindet, generiert. Alle diese Instanzen werden mit einem gleichen Typ „`Lookup`“ annotiert. Jede dieser Annotationen enthält gleichzeitig zwei Eigenschaften (*Features*), nämlich „class“ und „majorType“. Der Inhalt von „`class`“ ist der Konzeptname, den man in der Mapping-Datei angegeben hat. „`majorType`“ übernimmt den Inhalt von Feature1. Wenn Feature2 in der Indexdatei definiert wird, dann wird eine neue Eigenschaft „`minorType`“ der Annotation dazu erzeugt. Nach diesem Prozess werden bereits alle Instanzen der Ontologieklasse „SurfaceLevelEntity“ annotiert. Die Abbildung 4.2 zeigt das Resultat von einer unserer klinischen Studien.

JAPE Transducer bietet die flexible Möglichkeit, die Annotationen zu manipulieren [26]. Deshalb wird der erste JAPE Transducer unserer Anwendung für diesen Zweck verwendet. Nach dem Prozess der OntoGazetteer enthält jede Annotation nur die Standartfeatures. Obwohl die Annotationen die unterschiedlichen Typen haben, befinden sie sich nur in einem Annotation Set. Für den weiteren Prozess

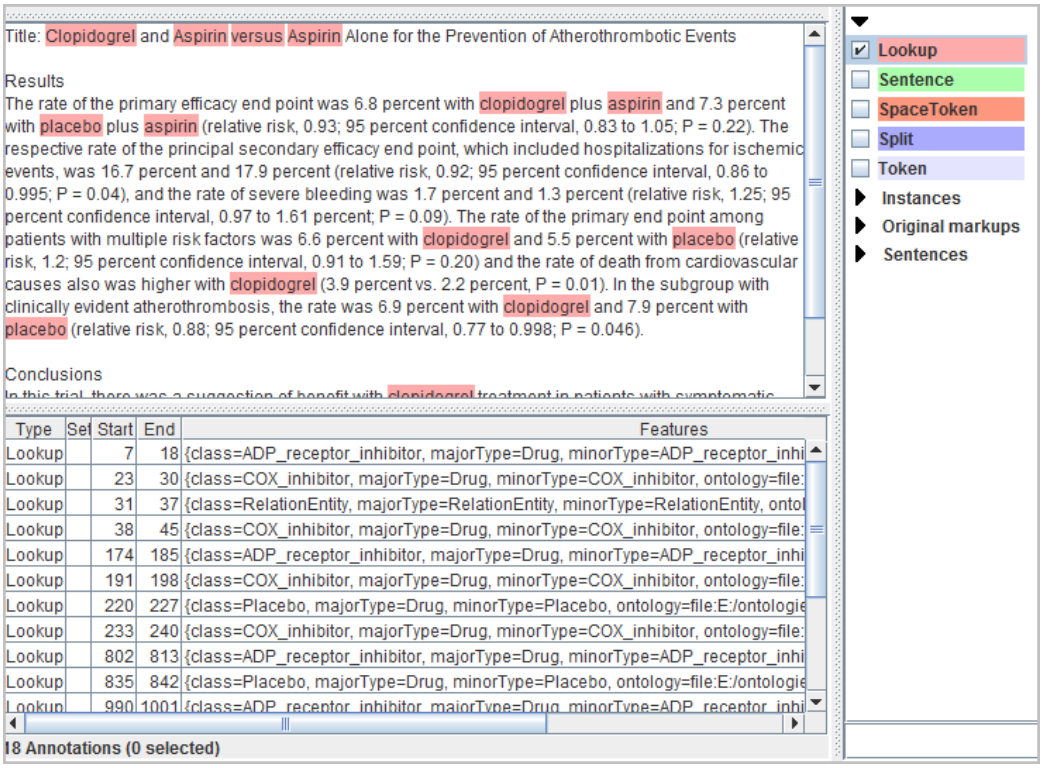

Abbildung 4.2: Resultat nach dem Prozess von OntoGazetteer

vom ABox Generator manipuliert ein JAPE Transducer diese Annotationen. Zum Beispiel wird eine Annotation des Medikamentes mit dem Typus „`instance`“ gekennzeichnet und dann ins neue Annotation Set „`Instances`“ hinzugefügt. Das Feature „`class`“ wird in „`ontoClass`“ umbenannt. Das Feature „`majorType`“ der gesuchten Entitäten hat den gleichen Wert „`cInstance`“. Die gesuchten Relationen werden ebenfalls „`majorType = oProperty`“ zugewiesen. Der originale Text stellt sich als Wert des neuen Features „`content`“ dar. Alle Annotationen behalten bei diesem Prozess ihre Informationen der Start- und Endposition.

Wie bereits im Abschnitt *Anwendungsszenario* erklärt, sollte das Textfragment „clopidogrel plus aspirin“ nicht als zwei Instanzen von Medikamenten betrachten werden, sondern als eine Instanz einer Kombination aus zwei Medikamenten. Dies gilt ebenfalls für das Textfragment „placebo plus aspirin“. Die folgende Regel im zweiten JAPE Transducer gibt uns die Lösung dieses Problems.

Listing 4.2: JAPE Regel drugcombination.jape

```
1  Input: Token Lookup
2
3  Rule: CombiDrugs
4  (
5          ({Lookup.majorType == Drug}):individual1
6          ({Token.string == "plus"}|{Token.string=="and"})
7          ({Lookup.majorType == Drug}):individual2
8  ):oneIndividual
9  -->
10 :oneIndividual{
11         gate.AnnotationSet aIndividual = (gate.
               AnnotationSet)bindings.get("oneIndividual");
12         gate.Annotation individualAnn = (gate.Annotation
               )aIndividual.iterator().next();
13         gate.FeatureMap features = Factory.newFeatureMap
               ();
14
15         int begOffset = aIndividual.firstNode().
               getOffset().intValue();
16         int endOffset = aIndividual.lastNode().getOffset
```

```
            ().intValue();
17          String mydocContent = doc.getContent().toString
            ();
18          String matchedString = mydocContent.substring(
            begOffset, endOffset);
19
20          features.put("ontoClass", "DrugCombination");
21          features.put("majorType", "cIinstance");
22          features.put("content", matchedString);
23
24          outputAS.add(aIndividual.firstNode(),
            aIndividual.lastNode(), "instance", features)
            ;
25 }
```

Zwei mit einem Wort „plus“ oder „and“ verbundene Medikamente werden die Ausführung dieser JAPE Regel auslösen. Dabei wird eine neue Instanz der Klasse „DrugCombination“ erzeugt. Diese Instanz hat die gleichen Eigenschaften wie die anderen Instanzen „`majorType`“, „`content`“ und „`ontoClass`“. Abbildung 4.3 zeigt das Ergebnis nach dem Prozess von JAPE Transducer.

Am Ende dieser Corpus Pipeline Applikation sind alle Instanzen in den Dokumenten bereit für die Generierung von ABoxen. Jedes Dokument im Korpus wird einer ABox zugeordnet. Obwohl GATE eine Ontologie API enthält, wird Jena Ontology API [10] in dieser Arbeit für die Generierung von ABox verwendet.

4.3 Interpretationsregel

Wir haben eine Gazetteer Liste mit möglichen Wörtern für die Vergleichsrelation verwendet. Bislang werden diese Wörter nur als einfache Individuen von RelationEntity in der Analyze-ABox identifiziert. In der ABox gibt es nur die Instanzen von SurfaceLevelEntity. Das folgende Diagramm zeigt die Instanzenstruktur dieses folgenden Satzes. `SEN-6` ist die Identifikation dieses Satzes in der ABox.

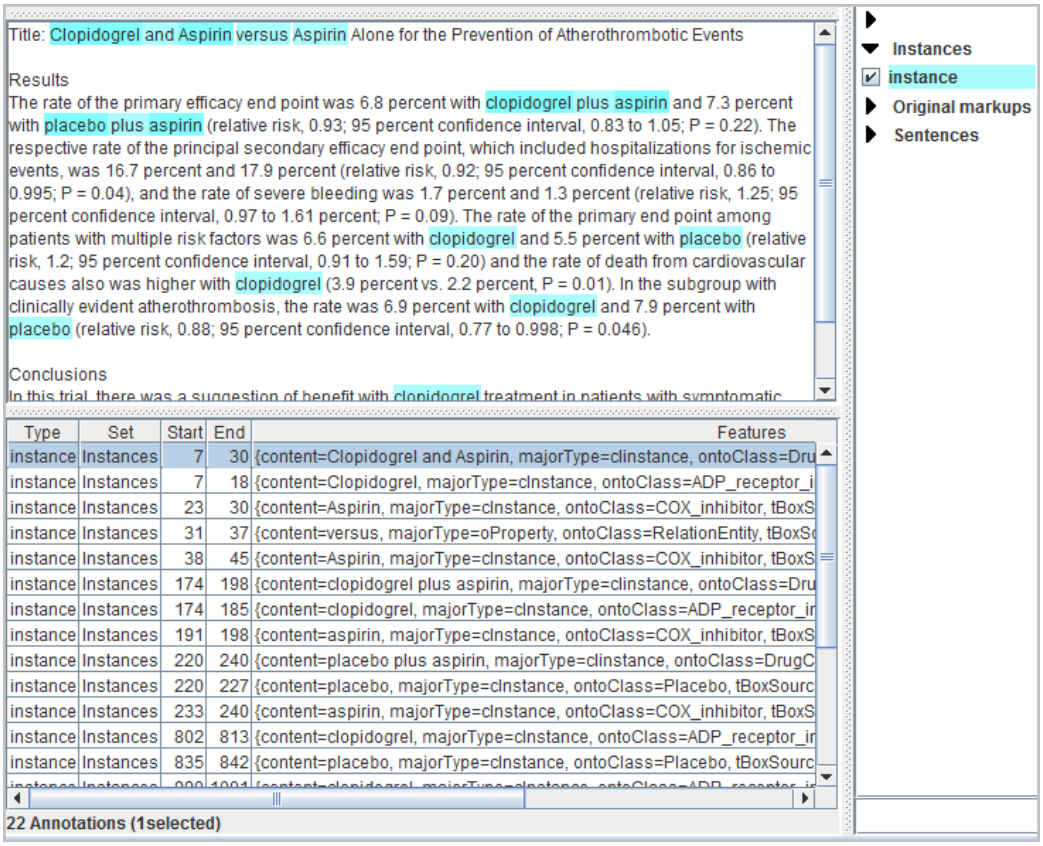

Abbildung 4.3: Resultat nach dem Prozess von JAPE Transducer

> Mean blood pressure was lower in both the telmisartan group (a 0.9/0.6 mm Hg greater reduction) and the combination-therapy group (a 2.4/1.4 mm Hg greater reduction) than in the ramipril group. [15]

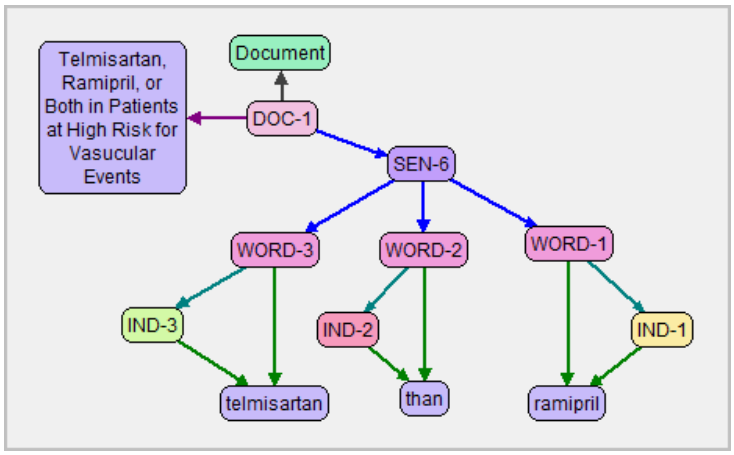

Abbildung 4.4: Instanzenstruktur des Beispielsatzes vor der Interpretation

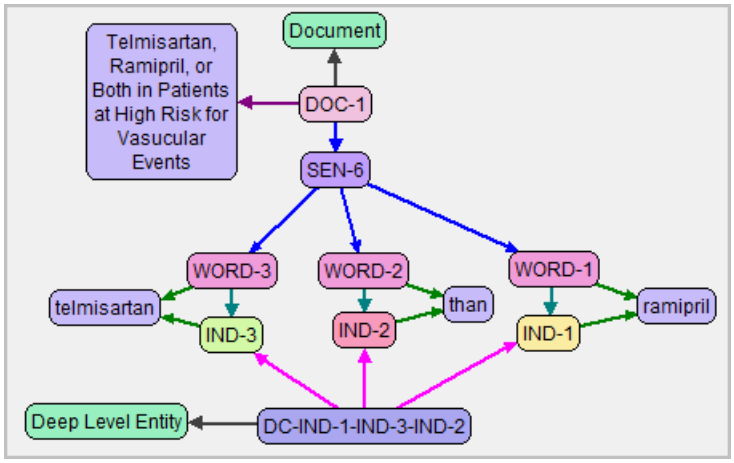

Abbildung 4.5: Instanzenstruktur des Beispielsatzes nach der Interpretation

Wie das Diagramm 4.4 uns zeigt, werden alle drei Instanzen von Medikamenten und der Vergleichsrelation nach der Informationsextraktion korrekt als Individuen in der Analyze-ABox identifiziert. Es gibt jedoch keine direkten semantischen Verbindungen zwischen den Instanzen in der ABox. Um diese Vergleichsrelation richtig zu implementieren, wird eine einfache Interpretationsregel in einer Inferenzmaschine definiert. Wenn zwei Drug Individuen und ein RelationEntity Individuum

in einem Satz vorkommen, wird diese Regel aktiviert. Eine neue Instanz der Klasse DrugComparison wird generiert, sie verbindet diese drei Individuen mit dem Property hasPart (siehe Abbildung 4.5). Natürlich ist diese Regel in obigem Beispiel sehr einfach definiert und nicht überall erfüllbar. Aber sie zeigt uns die Möglichkeit, dass sich das komplexe Wissen durch solche logische Regel einfach interpretieren lässt.

Gegenüber der JAPE-Regel ist die Interpretationsregel für komplexen Satzaufbau geeignet. Die zahlreichen Wörter und Symbole zwischen „telmisartan“ und „than“ in diesem Beispielsatz mit dem Regulärer-Ausdruck abzudecken, ist wesentlich komplizierter, als diese einfache Interpretationsregel zu schreiben.

4.4 Information-Retrieval

Das RDF (*Resource Description Framework*) ist eine formale Sprache für die Beschreibung strukturierter Informationen. Der TripleStore ist eine RDF-Datenbank. Triple bedeutet hier „Subjekt-Prädikat-Objekt“. Alle Einträge im TripleStore habewn diese Form. SPARQL (*SPARQL Protocol and RDF Query Language*) [13] bezeichnet eine Abfragesprache für die TripleStore, wie SQL für die normale Datenbank. Im Kern basiert SPARQL auf einfachen RDF-Anfragen in Form von Graph-Mustern. Darüber hinaus enthält SPARQL aber auch erweiterte Funktionen für die Konstruktion komplexerer Anfragemuster, für die Verwendung zusätzlicher Filterbedingungen und für die Formatierung der Ausgabe.

Eine SPARQL-Anfrage enthält drei wesentliche Bestandteile, gekennzeichnet durch die großgeschriebenen Schlüsselwörter `PREFIX`, `SELECT` und `WHERE`. Das Schlüsselwort `PREFIX` deklariert einen Namensraum. `SELECT` bestimmt das Ausgabeformat. Die Angaben nach SELECT sind die Bezeichnungen von Anfragevariablen, die wir als Rückgabewert anfordern möchten. Die eigentliche Anfrage wird durch das Schlüsselwort `WHERE` eingeleitet, Ihm folgt, eingeschlossen in geschweiften Klammern, ein einfaches Graph-Muster.

Das folgende Beispiel ist eine Anfrage in unserer Anwendung. Es wird nach den Titel, URL, Zeit und Ort der Veröffentlichung und den Verlag einer klinischen Studie gefragt. Als Bedingung soll das Dokument das Wort „aspirin“ enthalten.

Listing 4.3: Beispiel SPARQL, Suche nach Metainformationen von Dokumenten, die das Wort „aspirin“ enthalten.

```
PREFIX : <http://www.test.com/MedOnto.owl#>
PREFIX xsdt: <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?title ?sourceURL ?year ?publisher
WHERE {?doc  :hasTitle ?title;
             :sourceURL ?sourceURL;
             :publishYear ?year;
             :publishCity ?city;
             :publisherName ?publisher.
       ?word :depictsSLE ?drug;
             a :Word;
             :content "aspirin"^^xsdt:string.
       ?doc  :contains ?sentence.
       ?sentence :contains ?word.
      }
```

4.5 Bibliographie-Objekt

Word stellt hunderte von Objekten bereit, mit denen wir interagieren können. Diese Objekte werden in einer Hierarchie organisiert, die der Benutzeroberfläche eng folgt. Diese Hierarchie ist das sogenannte Word-Objektmodell. Das Microsoft Office Word 2007 Objektmodell enthält mehrere Typen von Objekten für die Erzeugung einer Bibliographie. Drei wichtige Objekttypen sind Source, Sources und Bibliography.

Source: eine individuelle Literaturressource wie Buch, Journalartikel oder Diplomarbeit usw.

Sources: Eine Kollektion von Source Objekten.

Bibliography: Die Liste von zitierter Sources in dem Dokument oder die Liste vorhandener Sources in der Applikation.

Wie in dem folgenden Beispiel zu sehen, wird jede Literaturressource in XML-Format repräsentiert.

```
<b:Source xmlns:b="http://schemas.microsoft.com/office/
    word/2004/10/bibliography">
    <b:Tag>And01</b:Tag>
    <b:SourceType></b:SourceType>
    <b:Author>
        <b:Author>
            <b:NameList>
                <b:Person>
                    <b:Last></b:Last>
                    <b:First></b:First>
                </b:Person>
            </b:NameList>
        </b:Author>
    </b:Author>
    <b:Title></b:Title>
    <b:Year></b:Year>
    <b:City></b:City>
    <b:Publisher></b:Publisher>
</b:Source>
```

Die Meta-Informationen in dieser XML sind die notwendigen Angaben einer Literaturressource. Als `SourceType` sind hier viele Typen von Literaturen hier erlaubt, zum Beispiel `Book`, `BookSection`, `JournalArticle`, `ConferenceProceedings` usw. In der NameList kann man mehrere Autoren angeben. All diesen Meta-Informationen werden bei der Informationsextraktionsphase in die ABox geschrieben. Bei Information-Retrieval werden diese Meta-Informationen aus der Wissensbasis als Suchergebnis in diesem XML Format an das Add-In-Programm in Word gesendet. Für die Referenzangabe in einem Word Dokument kann dann diese XML Datei direkt verwendet werden.

Kapitel 5

Zusammenfassung und Ausblick

In diesem Kapitel erfolgt eine Zusammenfassung der Arbeit. Im Anschluss daran werden in einem Ausblick mögliche Weiterentwicklungen diskutiert.

5.1 Zusammenfassung

Ziel der Arbeit war die Neuentwicklung eines Systems, das demonstriert, dass semantische Referenzen als Unterstützungen bei der Textverarbeitung möglich sind. Mit Hilfe der Ontologie wird domänspezifisches Wissen aus Texten extrahiert und interpretiert. Eine integrierte Komponente im Textverarbeitungsprogramm übermittelt dieses Wissen an den Anwender und unterstützt ihn bei der Zitierung.

Als erster Prototyp werden in dieser Arbeit zahlreiche vorhandene Tools und Frameworks für eine mögliche Zusammenarbeit untersucht. Mit Erfolg wird eine Systemarchitektur für unsere Aufgaben festgelegt. Für weitere spezifische Domänen sind Änderungen in der Phase der Informationsextraktion und Interpretation erforderlich. Mit vielen eingebauten NLP Komponenten ist GATE eine flexibles und leicht erweiterbares Framework für die Informationsextraktion. Fast ohne Änderung können die Komponenten wie Tokeniser und SentenceSplitter direkt wieder verwenden werden.

Die Trennung von Surface-Level- und Deep-Level-Entity erleichtet die Aufgaben

bei der ABox-Generierung und vermeidet die falsche Interpretation von Relationen. Mit Hilfe von LauguageUnitEntity werden die grundlegenden Informationen über die Dokumentstruktur in der ABox beibehalten. Für eine benutzerfreundliche Highlight-Markierung wäre diesen Informationen sehr sinnvoll. Auch für den Ansatz von des Maschine-Learning werden diesen Positionsinformationen benötigt.

Das Webservice sorgt für die gleiche Funktionalität selbst unterschiedlicher Anwendungsplattformen. Die semantische Referenzunterstützung kann dann in vielen Textverarbeitungsprogrammen integriert werden, zum Beispiel OpenOffice, Webbasierte Texteditor usw.

5.2 Ausblick

Mit der Implementierung dieser Arbeit wurde eine semantische Unterstützungsplattform für die Referenzierung bei der Textverarbeitung geschaffen. Diese Prototyp kann als Grundlage für die weitere Entwicklung genutzt werden. An einigen Stellen ist eine Anpassung und Erweiterung notwendig.

Momentan kann der Benutzer seine Abfrage lediglich in Form von Stichwörtern absenden. Statt mehrere Stichwörter als Suchbegriff zu benutzen, wäre eine Abfrage in der natürlichen Sprachform für den Anwender leichter nachzuvollziehen. Hierfür ist eine neue Prozess notwendig, der die Abfrage in der natürlichen Sprachform interpretieren kann.

Für die Referenzangaben werden die benötigten Meta-Informationen klinischer Studien zurzeit in einer Datei per Hand vorbereitet. Ein automatischer Extraktionsprozess der Meta-Informationen von Dokumenten ist als weitere Entwicklung dieser Arbeit sehr zu empfehlen.

Anhang A

TBox

Listing A.1: MedOnto.owl

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
    <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
    <!ENTITY MedOnto "http://www.test.com/MedOnto.owl#"
       >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml
       #" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#
       " >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-
       syntax-ns#" >
    <!ENTITY protege "http://protege.stanford.edu/
       plugins/owl/protege#" >
    <!ENTITY xsp "http://www.owl-ontologies.com
       /2005/08/07/xsp.owl#" >
]>

<rdf:RDF xmlns="http://www.test.com/MedOnto.owl#"
     xml:base="http://www.test.com/MedOnto.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
```

```
20      xmlns:protege="http://protege.stanford.edu/plugins/
           owl/protege#"
21      xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
22      xmlns:xsp="http://www.owl-ontologies.com
           /2005/08/07/xsp.owl#"
23      xmlns:owl="http://www.w3.org/2002/07/owl#"
24      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
25      xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
26      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
           ns#"
27      xmlns:MedOnto="http://www.test.com/MedOnto.owl#">
28    <owl:Ontology rdf:about="">
29        <rdfs:isDefinedBy rdf:datatype="&xsd;string">
             Qiang Li</rdfs:isDefinedBy>
30        <owl:versionInfo xml:lang="en">0.0.2</owl:
             versionInfo>
31        <rdfs:comment xml:lang="en"
32            >Diplomarbeit Test Ontology</rdfs:comment>
33    </owl:Ontology>
34
35    <!--
36    ///////////////////////////////////////////////////
37    // Object Properties
38    ///////////////////////////////////////////////////
39     -->
40    <!-- http://www.test.com/MedOnto.owl#contains -->
41    <owl:ObjectProperty rdf:about="#contains"/>
42
43    <!-- http://www.test.com/MedOnto.owl#depictsDLE -->
44    <owl:ObjectProperty rdf:about="#depictsDLE">
45        <rdfs:range rdf:resource="#DeepLevelEntity"/>
46        <rdfs:domain rdf:resource="#Document"/>
47    </owl:ObjectProperty>
48
49    <!-- http://www.test.com/MedOnto.owl#depictsSLE -->
50    <owl:ObjectProperty rdf:about="#depictsSLE">
51        <rdfs:range rdf:resource="#SurfaceLevelEntity"/>
52        <rdfs:domain rdf:resource="#Word"/>
53    </owl:ObjectProperty>
54
55    <!-- http://www.test.com/MedOnto.owl#hasAuthor -->
56    <owl:ObjectProperty rdf:about="#hasAuthor">
57        <rdfs:range rdf:resource="#Author"/>
```

```
58          <rdfs:domain rdf:resource="#Document"/>
59      </owl:ObjectProperty>
60
61      <!-- http://www.test.com/MedOnto.owl#hasPart -->
62      <owl:ObjectProperty rdf:about="#hasPart"/>
63
64      <!-- http://www.test.com/MedOnto.owl#isAbout -->
65      <owl:ObjectProperty rdf:about="#isAbout"/>
66
67      <!--
68      ///////////////////////////////////////////////////
69      // Data properties
70      ///////////////////////////////////////////////////
71       -->
72
73      <!-- http://www.test.com/MedOnto.owl#authorFirstName
            -->
74      <owl:DatatypeProperty rdf:about="#authorFirstName">
75          <rdfs:domain rdf:resource="#Author"/>
76          <rdfs:range rdf:resource="&xsd;string"/>
77      </owl:DatatypeProperty>
78
79      <!-- http://www.test.com/MedOnto.owl#authorLastName
            -->
80      <owl:DatatypeProperty rdf:about="#authorLastName">
81          <rdfs:domain rdf:resource="#Author"/>
82          <rdfs:range rdf:resource="&xsd;string"/>
83      </owl:DatatypeProperty>
84
85      <!-- http://www.test.com/MedOnto.owl#beginAt -->
86      <owl:DatatypeProperty rdf:about="#beginAt">
87          <rdfs:range rdf:resource="&xsd;long"/>
88          <rdfs:domain>
89              <owl:Class>
90                  <owl:unionOf rdf:parseType="Collection">
91                      <rdf:Description rdf:about="#
                            Sentence"/>
92                      <rdf:Description rdf:about="#Word"/>
93                  </owl:unionOf>
94              </owl:Class>
95          </rdfs:domain>
96      </owl:DatatypeProperty>
97
```

```
    <!-- http://www.test.com/MedOnto.owl#content -->
    <owl:DatatypeProperty rdf:about="#content">
        <rdfs:range rdf:resource="&xsd;string"/>
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#
                       SurfaceLevelEntity"/>
                    <rdf:Description rdf:about="#Word"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:DatatypeProperty>

    <!-- http://www.test.com/MedOnto.owl#endAt -->
    <owl:DatatypeProperty rdf:about="#endAt">
        <rdfs:range rdf:resource="&xsd;long"/>
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#
                       Sentence"/>
                    <rdf:Description rdf:about="#Word"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:DatatypeProperty>

    <!-- http://www.test.com/MedOnto.owl#hasTitle -->
    <owl:DatatypeProperty rdf:about="#hasTitle">
        <rdfs:domain rdf:resource="#Document"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://www.test.com/MedOnto.owl#publishCity -->
    <owl:DatatypeProperty rdf:about="#publishCity">
        <rdfs:domain rdf:resource="#Document"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://www.test.com/MedOnto.owl#publishYear -->
    <owl:DatatypeProperty rdf:about="#publishYear">
        <rdfs:domain rdf:resource="#Document"/>
```

```
139         <rdfs:range rdf:resource="&xsd;string"/>
140     </owl:DatatypeProperty>
141
142
143
144     <!-- http://www.test.com/MedOnto.owl#publisherName
            -->
145     <owl:DatatypeProperty rdf:about="#publisherName">
146         <rdfs:range rdf:resource="&xsd;string"/>
147     </owl:DatatypeProperty>
148
149     <!-- http://www.test.com/MedOnto.owl#sourceTypeValue
            -->
150     <owl:DatatypeProperty rdf:about="#sourceTypeValue">
151         <rdfs:range rdf:resource="&xsd;string"/>
152     </owl:DatatypeProperty>
153
154     <!-- http://www.test.com/MedOnto.owl#sourceURL -->
155     <owl:DatatypeProperty rdf:about="#sourceURL">
156         <rdfs:domain rdf:resource="#Document"/>
157         <rdfs:range rdf:resource="&xsd;string"/>
158     </owl:DatatypeProperty>
159
160     <!--
161     ///////////////////////////////////////////////////
162     // Classes
163     ///////////////////////////////////////////////////
164      -->
165
166     <!-- http://www.test.com/MedOnto.owl#ACE_inhibitor
            -->
167     <owl:Class rdf:about="#ACE_inhibitor">
168         <rdfs:subClassOf rdf:resource="#
                AntihypertensiveDrug"/>
169         <rdfs:comment rdf:datatype="&xsd;string"
170             >Angiotensin-converting enzyme</rdfs:comment
                >
171     </owl:Class>
172
173     <!-- http://www.test.com/MedOnto.owl#
            ADP_receptor_inhibitor -->
174     <owl:Class rdf:about="#ADP_receptor_inhibitor">
```

```
175         <rdfs:subClassOf rdf:resource="#AntiplateletDrug
               "/>
176         <rdfs:comment rdf:datatype="&xsd;string"
177             >Adenosine diphosphate(ADP) receptor
                   inhibitors</rdfs:comment>
178     </owl:Class>
179
180     <!-- http://www.test.com/MedOnto.owl#AT1_antagonist
           -->
181     <owl:Class rdf:about="#AT1_antagonist">
182         <rdfs:subClassOf rdf:resource="#
               AntihypertensiveDrug"/>
183         <rdfs:comment rdf:datatype="&xsd;string"
184             >Angiotensin II receptor type 1</rdfs:
                   comment>
185     </owl:Class>
186
187     <!-- http://www.test.com/MedOnto.owl#
           AntihypertensiveDrug -->
188     <owl:Class rdf:about="#AntihypertensiveDrug">
189         <rdfs:subClassOf rdf:resource="#Drug"/>
190     </owl:Class>
191
192     <!-- http://www.test.com/MedOnto.owl#
           AntiplateletDrug -->
193     <owl:Class rdf:about="#AntiplateletDrug">
194         <rdfs:subClassOf rdf:resource="#Drug"/>
195     </owl:Class>
196
197     <!-- http://www.test.com/MedOnto.owl#Author -->
198     <owl:Class rdf:about="#Author">
199         <rdfs:subClassOf rdf:resource="#MetaInfo"/>
200     </owl:Class>
201
202     <!-- http://www.test.com/MedOnto.owl#COX_inhibitor
           -->
203     <owl:Class rdf:about="#COX_inhibitor">
204         <rdfs:subClassOf rdf:resource="#AntiplateletDrug
               "/>
205         <rdfs:comment rdf:datatype="&xsd;string"
206             >Cyclooxygenase inhibitors</rdfs:comment>
207     </owl:Class>
208
```

```
    <!-- http://www.test.com/MedOnto.owl#DeepLevelEntity -->
    <owl:Class rdf:about="#DeepLevelEntity">
        <rdfs:subClassOf rdf:resource="#DomainEntity"/>
    </owl:Class>

    <!-- http://www.test.com/MedOnto.owl#Disease -->
    <owl:Class rdf:about="#Disease">
        <rdfs:subClassOf rdf:resource="#SurfaceLevelEntity"/>
    </owl:Class>

    <!-- http://www.test.com/MedOnto.owl#DiseaseOfTheCirculatorySystem -->
    <owl:Class rdf:about="#DiseaseOfTheCirculatorySystem">
        <rdfs:subClassOf rdf:resource="#Disease"/>
    </owl:Class>

    <!-- http://www.test.com/MedOnto.owl#Document -->
    <owl:Class rdf:about="#Document">
        <rdfs:subClassOf rdf:resource="#LinguisticUnit"/>
    </owl:Class>

    <!-- http://www.test.com/MedOnto.owl#DomainEntity -->
    <owl:Class rdf:about="#DomainEntity"/>

    <!-- http://www.test.com/MedOnto.owl#Drug -->
    <owl:Class rdf:about="#Drug">
        <rdfs:subClassOf rdf:resource="#SurfaceLevelEntity"/>
    </owl:Class>



    <!-- http://www.test.com/MedOnto.owl#DrugCombination -->
    <owl:Class rdf:about="#DrugCombination">
        <rdfs:subClassOf rdf:resource="#SurfaceLevelEntity"/>
    </owl:Class>
```

```
243
244     <!-- http://www.test.com/MedOnto.owl#DrugComparison
            -->
245     <owl:Class rdf:about="#DrugComparison">
246         <rdfs:subClassOf rdf:resource="#DeepLevelEntity"
                />
247     </owl:Class>
248
249     <!-- http://www.test.com/MedOnto.owl#Hypertension
            -->
250     <owl:Class rdf:about="#Hypertension">
251         <rdfs:subClassOf rdf:resource="#
                DiseaseOfTheCirculatorySystem"/>
252     </owl:Class>
253
254     <!-- http://www.test.com/MedOnto.owl#LinguisticUnit
            -->
255     <owl:Class rdf:about="#LinguisticUnit"/>
256
257     <!-- http://www.test.com/MedOnto.owl#MetaInfo -->
258     <owl:Class rdf:about="#MetaInfo"/>
259
260     <!-- http://www.test.com/MedOnto.owl#
            Myocardial_Infarction -->
261     <owl:Class rdf:about="#Myocardial_Infarction">
262         <rdfs:subClassOf rdf:resource="#
                DiseaseOfTheCirculatorySystem"/>
263     </owl:Class>
264
265     <!-- http://www.test.com/MedOnto.owl#Placebo -->
266     <owl:Class rdf:about="#Placebo">
267         <rdfs:subClassOf rdf:resource="#
                SurfaceLevelEntity"/>
268     </owl:Class>
269
270     <!-- http://www.test.com/MedOnto.owl#RelationEntity
            -->
271     <owl:Class rdf:about="#RelationEntity">
272         <rdfs:subClassOf rdf:resource="#
                SurfaceLevelEntity"/>
273     </owl:Class>
274
275     <!-- http://www.test.com/MedOnto.owl#Sentence -->
```

```
    <owl:Class rdf:about="#Sentence">
        <rdfs:subClassOf rdf:resource="#LinguisticUnit"
           />
    </owl:Class>

    <!-- http://www.test.com/MedOnto.owl#
       SurfaceLevelEntity -->
    <owl:Class rdf:about="#SurfaceLevelEntity">
        <rdfs:subClassOf rdf:resource="#DomainEntity"/>
    </owl:Class>

    <!-- http://www.test.com/MedOnto.owl#Word -->
    <owl:Class rdf:about="#Word">
        <rdfs:subClassOf rdf:resource="#LinguisticUnit"
           />
    </owl:Class>
</rdf:RDF>
<!-- Generated by the OWL API (version 2.2.1.941) http:
   //owlapi.sourceforge.net -->
```

Anhang B

Textdokumente für Implementierung

Für die Implementierung werden folgende Abschnitten aus vier klinischen Studien verwendet. Die originalen Studien in Vollvision finden Sie in der angegebenen Literaturliste.

B.1 Textdokument 1

[15] ***Title*** Telmisartan, Ramipril, or Both in Patients at High Risk for Vascular Events

Results Mean blood pressure was lower in both the telmisartan group (a 0.9/0.6 mm Hg greater reduction) and the combination-therapy group (a 2.4/1.4 mm Hg greater reduction) than in the ramipril group. At a median follow-up of 56 months, the primary outcome had occurred in 1412 patients in the ramipril group (16.5%), as compared with 1423 patients in the telmisartan group (16.7%; relative risk, 1.01; 95% confidence interval [CI], 0.94 to 1.09). As compared with the ramipril group, the telmisartan group had lower rates of cough (1.1% vs. 4.2%, P<0.001) and angioedema (0.1% vs. 0.3%, P = 0.01) and a higher rate of hypotensive symptoms (2.6% vs. 1.7%, P<0.001); the rate of syncope was the same in the two groups (0.2%). In the combination-therapy group, the primary outcome occurred in 1386 patients (16.3%; relative risk, 0.99; 95% CI, 0.92 to 1.07); as compared with the

ramipril group, there was an increased risk of hypotensive symptoms (4.8% vs. 1.7%, P<0.001), syncope (0.3% vs. 0.2%, P = 0.03), and renal dysfunction (13.5% vs. 10.2%, P<0.001).

Conclusions Telmisartan was equivalent to ramipril in patients with vascular disease or highrisk diabetes and was associated with less angioedema. The combination of the two drugs was associated with more adverse events without an increase in benefit.

B.2 Textdokument 2

[2] ***Title*** Effects of the angiotensin-receptor blocker telmisartan on cardiovascular events in high-risk patients intolerant to angiotensin-converrting enzyme inhibitors: a randomised controlled trial

Findings The median duration of follow-up was 56(IQR 51-64) months. All randomised patients were included in the efficacy analyses. Mean blood pressure was lower in the telmisartan group than in the placebo group throughout the study (weighted mean difference between group 4.0/2.2 [SD 19.6/12.0]mm Hg). 465(15.7%) patients experienced the primary out come in the telmisartan group compared with 504 (17.0%) in the placebo group (hazard ratio 0.92, 95% CI 0.81-1.05, p=0.216). One of the secondary outcomes - acomposite of cardiovascular death, myocardial infarctionm or stroke - orccureed in 384 (13.0%) patients on telmisartan compared with 440 (14.8%) on placebo (0.87, 0.76-1.00, p=0.048 unadjusted; p=0.068 after adjustment for multiplicity of comparisons and overlap with primary outcome). 894(30.3%) patient receiving telmisartan were hospitalised for a cardiovascular reason, compared with 980 (33.0%) on placebo (relative risk 0.92, 95% CI 0.85-0.99; p=0.025). Fewer patients permanently discontnued study medication in the telmisartan group than in the placebo group (639[21.6%] vs 705[23.8%]; p=0.055); the most common reason for permanent discontinuation as hypotensive symptoms(29[0.98%] in the telmisartan group vs 16[0.54%] in the placebo group).

Interpretation Telmisartan was well tolerated in patients unable to tolerate ACE inhibitors. Although teh drug had no significatn effect on the primary outcome of

this study, which included hospitalisations for heart failure, it modestly reduced the risk of the composite outcome of cardiovascular death, myocardial infarction or stroke.

B.3 Textdokument 3

[1] ***Title*** A randomised, blinded, trial of clopidogrel versus aspirin in patients at risk of ischaemic events (CAPRIE)

Findings 19185 patients, with more than 6300 in each of the clinical subgroups, were recruited over 3 years, with a mean follow-up of 1.91 years. There were 1960 first events included in the outcome cluster on which an intention-totreat analysis showed that patients treated with clopidogrel had an annual 5.32% risk of ischaemic stroke, myocardial infarction, or vascular death compared with 5.83% with aspirin. These rates reflect a statisticaly significant (p=0.043) relative-risk reduction of 8.7% in favour of clopidogrel (95% CI 0.3–16.5). Corresponding on-treatment analysis yielded a relative-risk reduction of 9.4%. There were no major differences in terms of safety. Reported adverse experiences in the clopidogrel and aspirin groups judged to be severe included rash (0.26% vs 0.10%), diarrhoea (0.23% vs 0.11%), upper gastrointestinal discomfort (0.97% vs 1.22%), intracranial haemorrhage (0.33% vs 0.47%), and gastrointestinal haemorrhage (0.52% vs 0.72%), respectively. There were ten (0.10%) patients in the clopidogrel group with significant reductions in neutrophils (<1.2 3 109/L) and 16 (0.17%) in the aspirin group.

Interpretation Long-term administration of clopidogrel to patients with atherosclerotic vascular disease is more effective than aspirin in reducing the combined risk of ischaemic stroke, myocardial infarction, or vascular death. The overall safety profile of clopidogrel is at least as good as that of medium-dose aspirin.

B.4 Textdokument 4

[6] ***Title*** Clopidogrel and Aspirin versus Aspirin Alone for the Prevention of Atherothrombotic Events

Results The rate of the primary efficacy end point was 6.8 percent with clopidogrel plus aspirin and 7.3 percent with placebo plus aspirin (relative risk, 0.93; 95 percent confidence interval, 0.83 to 1.05; P = 0.22). The respective rate of the principal secondary efficacy end point, which included hospitalizations for ischemic events, was 16.7 percent and 17.9 percent (relative risk, 0.92; 95 percent confidence interval, 0.86 to 0.995; P = 0.04), and the rate of severe bleeding was 1.7 percent and 1.3 percent (relative risk, 1.25; 95 percent confidence interval, 0.97 to 1.61 percent; P = 0.09). The rate of the primary end point among patients with multiple risk factors was 6.6 percent with clopidogrel and 5.5 percent with placebo (relative risk, 1.2; 95 percent confidence interval, 0.91 to 1.59; P = 0.20) and the rate of death from cardiovascular causes also was higher with clopidogrel (3.9 percent vs. 2.2 percent, P = 0.01). In the subgroup with clinically evident atherothrombosis, the rate was 6.9 percent with clopidogrel and 7.9 percent with placebo (relative risk, 0.88; 95 percent confidence interval, 0.77 to 0.998; P = 0.046).

Conclusions In this trial, there was a suggestion of benefit with clopidogrel treatment in patients with symptomatic atherothrombosis and a suggestion of harm in patients with multiple risk factors. Overall, clopidogrel plus aspirin was not significantly more effective than aspirin alone in reducing the rate of myocardial infarction, stroke, or death from cardiovascular causes.

Anhang C

Interpretationsregel

In der Interpreationsphase wird folgende Regel verwendet.

```
(firerule (and (?x #!medOnt:Document)
    (?x ?y #!medOnt:contains) (?y #!medOnt:Sentence)
    (?y ?w1 #!medOnt:contains) (?w1 #!medOnt:Word)
    (?y ?w2 #!medOnt:contains) (?w2 #!medOnt:Word)
    (?y ?w3 #!medOnt:contains) (?w3 #!medOnt:Word)
    (?w1 ?drug1 #!medOnt:depictsSLE)
    (?w2 ?drug2 #!medOnt:depictsSLE)
    (?w3 ?rel #!medOnt:depictsSLE)
    (neg (same-as ?drug1 ?drug2))
    (?drug1 #!medOnt:Drug)
    (?drug2 #!medOnt:Drug)
    (?rel #!medOnt:RelationEntity))
   ((instance (new-ind DC ?drug1 ?drug2 ?rel) #!medOnt:
      DrugComparison)
    (related ?x (new-ind DC ?drug1 ?drug2 ?rel) #!medOnt
       :depictsDLE)
    (related (new-ind DC ?drug1 ?drug2 ?rel) ?drug1 #!
       medOnt:hasPart)
    (related (new-ind DC ?drug1 ?drug2 ?rel) ?drug2 #!
       medOnt:hasPart)
    (related (new-ind DC ?drug1 ?drug2 ?rel) ?rel #!
       medOnt:hasPart)
    (related ?x (new-ind DC ?drug1 ?drug2 ?rel) #!medOnt
       :isAbout)))
```

Literaturverzeichnis

[1] *A randomised, blinded, trial of clopidogrel versus aspirin in patients at risk of ischaemic events (CAPRIE). CAPRIE Steering Committee.* Lancet, http://view.ncbi.nlm.nih.gov/pubmed/8918275, 348(9038):1329–1339, November 1996.

[2] *Effects of the angiotensin-receptor blocker telmisartan on cardiovascular events in high-risk patients intolerant to angiotensin-converting enzyme inhibitors: a randomised controlled trial.* Lancet, August 2008.

[3] Appelt, Douglas E. und David J. Israel: *Introduction to Information Extraction Technology.* AI Communications 12(3), 12(3), 1999.

[4] Baader, F., D. Calvanese, D.L. McGuinness, D. Nardi und P.F. Patel-Schneider: *The Description Logics Handbook: Theory, Implementations, and Applications.* Cambridge University Press, 2003.

[5] Bechhofer, Sean, Frank van Harmelen, Jim Hendler, lan Horrocks, Deborah L. McGunness, Peter F. Patel-Schneider und Lynn Andrea Stein: *OWL Web Ontology Lanugage Reference.* http://www.w3.org/TR/owl-ref/#Property, 2004.

[6] Bhatt, Deepak L., Keith A.A. Fox, Werner Hacke, Peter B. Berger, Henry R. Black, William E. Boden, Patrice Cacoub, Eric A. Cohen, Mark A. Creager, J. Donald Easton, Marcus D. Flather, Steven M. Haffner, Christian W. Hamm, Graeme J. Hankey, S. Claiborne Johnston, Koon-Hou Mak, Jean-Louis Mas, Gilles Montalescot, Thomas A. Pearson, P. Gabriel Steg, Steven R.

Steinhubl, Michael A. Weber, Danielle M. Brennan, Liz Fabry-Ribaudo, Joan Booth, Eric J. Topol und the CHARISMA Investigators: *Clopidogrel and Aspirin versus Aspirin Alone for the Prevention of Atherothrombotic Events.* N Engl J Med, http://content.nejm.org/cgi/content/abstract/354/16/1706, 354(16):1706–1717, 2006.

[7] Chinchor, Nancy A.: *Overview Of MUC-7/MUC-2.* In: *In proceedings of the seventh Machine Understanding Conference*, 1998.

[8] Cunningham, H.: *Information Extraction, Automatic.* Encyclopedia of Language and Linguistics, 2nd Edition, 2005.

[9] Cunningham, Hamish., Diana Mayndard, Kalina Bontcheva, Valentin Tablan, Cristina Ursu, Marin Dimitrov, Mike Dowman, Niraj Aswani, Ian Roberts, Yaoyong Li, Andrey Shafirin und Adam Funk: *Developing Language Processing Component with GATE Version 4 (a User Guide).* The University of Sheffied, 2008.

[10] Dickinson, Ian: *Jena Ontology API.* http://jena.sourceforge.net/ontology/index.html, 01 2008.

[11] Grishman, R.: *TIPSTER Architecture Design Document Version 2.3.* Technischer Bericht, DARPA, 1997.

[12] Grishman, Ralph: *Information Extraction: Techniques and Challenges.* In: *SCIE '97: International Summer School on Information Extraction*, Seiten 10–27, London, UK, 1997. Springer-Verlag.

[13] Hitzler, Pascal, Markus Krötzsch, Sebastian Rudolph und York Sure: *Semantic Web Grundlagen.* Springer-Verlag, 2008.

[14] Horrocks, I. und F. van Harmelen: *Reference Description of DAML+OIL Ontology Markup Language.* Technischer Bericht, DARPA Agent Markup Language Program, March 2001.

[15] Investigators, The ONTARGET: *Telmisartan, Ramipril, or Both in Patients at High Risk for Vascular Events.* N Engl J Med, http://content.nejm.org/cgi/content/abstract/358/15/1547, 358(15):1547–1559, 2008.

[16] Kenter, Tom und Diana Maynard: *Using GATE as an Annotation Tool.* http://www.gate.ac.uk/sale/am/annotationmanual.pdf, 2005.

[17] Knublauch, Holger, Ray W. Fergerson, Natalya F. Noy und Mark A. Musen: *The Protégé OWL plugin: An open development environment for semantic web applications.* Springer, 2004.

[18] Lassila, O. und R. Swick: *Resource Description Framework (RDF) Model and Syntax Specification.* Technischer Bericht, W3C Consortium.

[19] Lorenzen, Klaus F.: *Zitieren und Belegen in wissenschaftlichen Arbeiten.* http://www.bui.haw-hamburg.de/pers/klaus.lorenzen/ASP/zitierenbelegen.pdf, 2003.

[20] M. Dean, G. Schreiber, S. Bechhofer F. van Harmelen J. Hendler I. Horrocks D. L. McGuinness P. F. Patel-Schneider und L. A. Stein.: *OWL web ontology language reference.* Technischer Bericht, W3C recommendation, Feb 2004.

[21] McGunness, Deborah L. und Frank van Harmelen: *OWL Web Ontology Lanugage Overview.* http://www.w3.org/TR/owl-features, 2004.

[22] McQuay, H und R Moore: *Placebo.* Postgraduate Medical Journal, 81, 2005.

[23] Monadjemi, Peter und Pfeifer Eckehard: *Microsoft Office 2007 - Programmierung – Das Entwicklerbuch.* Microsoft Press Deutschland, 2008.

[24] Sipser, Michael: *Introduction to the Theory of Computation.* International Thomson Publishing, 1996.

[25] Smith, Michael K., Chris Welty und Deborah L. McGuiness: *OWL Web Ontology Lanugage Guide.* http://www.w3.org/TR/owl-guide, 2004.

[26] Tablan, Valentin, Daina Maynard, Kalina Bontcheva und Hamish Cunningham: *GATE – An Application Developer's Guide.* http://gate.ac.uk/sale/pg/pg.pdf, 2004.

[27] Trill, R.: *Vom Krankenhausinformationssystem zum elektronischen Gesundheitswesen (eHealth).* FH Flensburg, Mai 2006.

[28] WIKIPEDIA: *Klinische Studie*. http://de.wikipedia.org/wiki/Klinische_Studie, 2008.