Institut für Rechnergestützte Ingenieursysteme

Fakultät Informatik, Elektrotechnik und Informationstechnik

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Diplomarbeit Nr. 3209

Implementierung von Methoden zur Auswertung von textuellen Anforderungen im WRSPM-Umfeld

Muhammed Tuncer

Studiengang:	INFORMATIK
Prüfer:	Univ-Prof. Hon-Prof. Dr. Dieter Roller
Betreuer:	Dipl. –Inf. Akram Chamakh
begonnen am:	03.08.11
beendet am:	02.02.12
CR-Klassifikation:	F.2.2, H.5.2, J.6

Abstract

In a competitive market of corporate software development, the numbers of important factors are increasing. In addition having a successful conduction for requirement engineering plays a significant role. Therefore supplier and customer have to contribute in the early phases of system development. In this way many errors can be eliminated. Linguistic methods can help to accelerate this process with the aid of computer systems.

The objective of this thesis is to generate a semantically annotated SALE-file (Semantic Annotation Language for English) from a pure text. SALE-file is a structured text file with semantic roles. To automatically produce the precise meaning of a text, the collaboration among syntactic structure with knowledge resources (ontology) was studied and implemented. Through a methodic unification of several similar ontologies, the exact results are obtained. For the realization of the concept and to visualize the results, a prototype called Auto Semantic Role Labeler was developed which uses the syntactic information supplier Stanford Core NLP in addition to WordNet, VerbNet and FrameNet ontologies.

Meanwhile the large projects like Eclipse-EMF can load and graphically display eSALE file (an XML file obtained from a SALE file). The results correspond to the instances (word.java, subphrase.java etc.) of the text. By loading this XML file the JAVA classes are obtained as well to be applied for further software development phases. [KIT11a], [KIT11b]

Zusammenfassung

In einem wettbewerbsorientierten Markt spielen bei der Entwicklung von Firmen-Software immer mehr Faktoren eine wichtige Rolle. Auch eine erfolgreich durchgeführte Anforderungsanalyse (bzw. Requierement Engineering) zählt dazu. Denn so können z.B. Auftraggeber und Anwender entscheidend in den frühen Phasen der Systementwicklung mitwirken. Viele Fehler, die erst später auftauchen, können somit von Anfang an beseitigt werden. Linguistische Methoden können helfen, diesen Prozess rechnergestützt zu beschleunigen.

Die Aufgabe dieser Arbeit ist es, aus einem reinen Text eine semantisch annotierte SALE-Datei (Semantic Annotation Language for English) zu erzeugen. Eine SALE-Datei ist eine strukturierte Text-Datei mit semantischen Rollen. Basierend auf einer syntaktischen Struktur wird in Zusammenarbeit mit Wissens-Ressourcen (Ontologien) eine möglichst genaue Bedeutung des Textes maschinell erarbeitet. Durch die methodische Vereinigung mehrerer Ontologien, die ähnlich aufgebaut sind, kommt man grundsätzlich auf exaktere Ergebnisse. Deswegen kamen in dem hier neu entwickelten Prototyp „Auto SRL“, neben dem syntaktischen Informationslieferant Stanford Core NLP noch folgende Ontologien zum Einsatz: WordNet, VerbNet, und FrameNet.

Es gibt mittlerweile große Projekte wie Eclipse-EMF, die ein strukturiertes Dokument wie z.B. eine eSALE-Datei (gewonnene XML-Datei aus einer SALE-Datei) laden und grafisch anzeigen können. Die Ergebnisse entsprechen dabei den Instanzen des Textes. Durch das Laden dieser XML-Datei erhält man gleichzeitig Java-Klassen, mit denen man im weiteren Vorgehen der Software-Entwicklung interagieren kann (siehe [KIT11a] und [KIT11b]).

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AA	AutoAnnotator
ADVP	Adverbphrase
Auto SRL	Auto Semantic Role Labeler
API	Application Programming Interface
BNF	Backus Nour Form
CIM	Computer-integrated-manufacturing
CL	Computer Linguistik
CR	Coneptual Relation
EE	Event Extraction
HG	Hypergrafen
IE	Information Extraction
IR	Information Retrieval
ISO	International Organization for Standardization
KAOS	Knowledge Acquisition in automated Specification
KIF	Knowledge Interchange Format
LSR	Lexical-Semantic Relation
LU	Lexical Unit
MDA	Model Driven Architecture
ME	Model Extraction
MX	Model Extractor
NER	Name Entity Recognition
NLP	Natural Language Processing
OG	Omnigraf
OMG	Object Management Group
OWL	Web Ontology Language

PAS	Prädikat-Argument-Strukturen
RDF	Ressource Description Framework
RDFS	Ressource Description Framework Schema
SALE	Semantic Annotation Language for English
SENSE	The Software Engineer's Natural-language Semantics Encoding
SRL	Semantic Role Labeling
SUMO	Suggested Upper Merged Ontology
SUO-KIF	Standard Upper Ontology Knowledge Interchange Format
Synset	Synset
TM	Topic Maps
Tt	Tentakel
UML	Unified Modeling Language
WRSPM	World Requirements Specification Program Machine
XMI	XML Metadata Interchange
XML	Extensible Markup Language

1 Einführung

Im Rahmen der Anforderungsanalyse des Automotiv-Bereichs wurden einige Methoden zur Verwaltung von Anforderungen entwickelt. KAOS, WRSPM (World Requirements Specification Program Machine) und andere Methoden versuchen Anforderungen zu formalisieren. Neben der Verwaltung spielen hauptsächlich linguistische Aspekte eine wichtige Rolle. Die meisten Anforderungen sind nämlich in reiner Textform meistens in Pflichten- und Lastenheften abgelegt, sodass diese nicht automatisch in den Softwareentwicklungsprozess einfließen können. Es bedarf einer effizienten Struktur-Speicherung und Modellierung von jeglichen Informationen sowie einer anschließenden Abbildung in die Software. Dadurch können kleine oder große Änderungen des Kunden viel schneller im Entwicklungsprozess berücksichtigt und realisiert werden, was ohne ein solches System zeitaufwändiger ist. Natürlich sind unstrukturierte Texte aufgrund keiner gegebenen einheitlichen Datenstruktur mit großer Problematik verbunden. Diese Probleme können nicht alle behoben werden, da die Grammatik der natürlichen Sprache unbeschränkt ist und somit diese Probleme nicht entscheidbar macht. Eine naheliegende Lösung wäre das Vorhandensein von sprachlichen Informationen, die fasst ausreichen, um einen fremden Text zu verstehen. Solche Informationen sind in Ontologien verfügbar. Deren Verknüpfung in der NLP (Natural Language Processing) bzw. hier in der Anforderungsanalyse versprechen, mit der Voraussetzung, dass sie nicht mehrdeutige Informationen liefern, viel. Die hier verwendeten Ontologien liefern wirklich viel Information, wobei diese Informationen dann manchmal nicht wirklich helfen, wenn sie mehrdeutig sind. Die größte Herausforderung bei der Kombination stellt dabei die wirklich nützlichen Informationen dabei aus zu picken oder so effektiv vorzugehen, dass nur noch nützliche Daten ausgegeben werden.

1.1 Problemstellung

Seit mehreren Jahren beschäftigt sich die Linguistik mit dem Problem des Verstehens von gegebenen Sätzen. Je komplizierter und länger ein Satz ist, desto problembehafteter ist ihre automatische Erkennung. Selbst bewährte Tools wie Stanford Parser zeigen hierbei manchmal ihre Schwächen, wobei diese auch nicht unterschätzt werden dürfen.

1.2 Ziel und Zweck der Diplomarbeit

In dieser Arbeit soll gezeigt werden, wie die Kombination mehrerer solcher Komponenten wie Stanford einen Vorteil verschaffen könnten. Damit kann auch gleichzeitig ermittelt werden, wie dieses hier erworbene Know-How bezüglich rechnergestützte Bearbeitung von natürlichen Texten zukünftig in der Analyse von Anforderungstexten eingesetzt werden kann.

2 Natural Language Processing

In diesem Kapitel werden zuerst linguistische Grundlagen besprochen. In [Mitk03] sind einige Grundlagen erwähnt, von denen hier nur Syntax, Semantik und Lexikografie behandelt werden. Anschließend werden Methoden sowie Anwendungen der Natural Language durchgenommen.

2.1 Grundlagen

Das Ziel einer syntaktischen und semantischen Analyse ist es, einen gegebenen Text mittels der Methoden der Computerlinguistik in kleinste Einheiten (Morphs) zu zerlegen und dann diese sowohl mit syntaktischen als auch mit semantischen Annotationen zu versehen.

2.1.1 Syntaktische Analyse

Eine syntaktische Analyse erfolgt mithilfe eines Parsers. Der Parser stellt die syntaktische Struktur eines Textes dar. Eine geeignete syntaktische Struktur ist für die Qualität der semantischen Analyse, letztendlich für das System, entscheidend. Eine effektive Struktur ist aber aufgrund der Komplexität der natürlichen Sprache nicht ganz möglich. Zielverwandte Arbeiten zeigen, dass sie meistens aufgrund der gerade erwähnten Komplexität den Sprachumfang einschränken müssen, um dadurch das gewünschte Ergebnis zu erzielen.

2.1.2 Semantische Analyse

Für die Semantik kann in der CL Umgebung bekannte Prädikat-Argument-Struktur eingesetzt werden. Daneben gibt es nach [Gelh10] andere Ansätze, die zur semantischen Analyse und als Grundlage für die Modellbildung sich eventuell besser eignen. All diese Ansätze werden im Kapitel 2.4 näher erläutert.

Bei den natürlichen Sprachen ist es sinnvoll direkt anhand der Semantik-Struktur ein Strukturmodell zu bauen. Durch Syntax entstände nämlich eine n:m Beziehung. Das bedeutet, dass eine auf der Grundlage von Wortfolgen (bzw. aus der syntaktischen Struktur) gebildetes Modell mehrdeutig ist und umgekehrt [Gelh10, S. 109]. Das Ziel der semantischen Analyse ist es, die semantischen Rollen den Tokens zu zuweisen und in unserem Fall eine Grundlage für die Modellierung der Entitäten sowie ihre Verknüpfungen zu schaffen.

2.1.3 Lexikografie und semantische Netzwerke

Mittlerweile gibt es eine Vielzahl von lexikalischen Ressourcen bzw. Datenbanken, die bei der Verarbeitung von natürlichen Sprachen zum Einsatz kommen. In manchen Literaturen werden Ressourcen als Ontologien bezeichnet, da wahrscheinlich einige Ontologie-Konzepte darin enthalten sind. Die Ressourcen werden z.B. bei den Methoden Parsing, Tagging und Ontologie etc. eingesetzt. Im Bereich der natürlichen Sprachen zählen WordNet, **V**erbNet, **P**robBank,

FrameNet, **O**nto Notes Sense Groupings zu den wichtigen Datenbeständen. **VPFO** ist eine vereinigte Datenbank, die all die obigen DBs außer der WordNet enthält. Nun werden exemplarisch einige Wort-Datenbanken nämlich Frame-, Word-, Verb- und GermaNet vorgestellt, die im Verlauf dieser Arbeit mehr oder weniger eingesetzt wurden.

FrameNet basiert auf eine Frame-Semantic und verwaltet seine Frames anhand einer Indexstruktur. Das Wort fried bzw. fry.v taucht im Frame Apply Heat in der Tabelle eins als Lexical Unit (LU) auf. Diese LU kann FrameElemente (FE) innerhalb des gleichen Frames hervorrufen bzw. auslösen. Die Auslösung gleicher FE kann auch durch die andere LU bake.v erfolgen. FrameNet DB enthält ca. 1000 indexierte Frames.

FrameSQL ist eine webbasierte Anwendung, um eine strukturierte Suche in der FrameNet DB durchzuführen. Neue Features dieses Tools weißen Mehrsprachigkeit in den Sprachen Spanisch, Japanisch und in Deutsch auf.

Tabelle 1: Das Frame Apply_Heat in FrameNet

Definition: Beschreibung des Frames FE: Food, Container, Heating_Instrument, Cook LE: fry.v, bake.v

LU's, die in der unteren Tabelle vorkommen, sind Prädikate. Hier werden z.B. drei Frame-Elemente (Cook, Food, Heating_Instrument) durch das Prädikat fried ausgelöst. LU’s in FrameNet sind meistens Prädikate bzw. Verben oder substantivierte Verben oder Adjektive.

Tabelle 2: Prädikat als Lexical Unit

Matilde	**fried**	the catfish	in a heavy iron skillet
Cook		*Food*	*Heating_Instrument*

Tabelle 3: Substantiviertes Verb als Lexical Unit

…	the	**reduction**	of debt levels
			Item

Tabelle 4: Adjektiv als Lexical Unit

They	were	**asleep**	for hours
Sleeper	*Copulo*		*Duration*

WordNet basiert gegenüber FrameNet nicht auf Frame-Struktur. Es besitzt eine Wortstruktur. Dabei zeigt es semantische Beziehungen zwischen den Wörtern an. Das Synset in der unteren Tabelle ist ein Kürzel in WordNet, das alle Synonyme zum betrachteten Wort (LU) sowohl als Substantiv als auch als Verb enthält [Word11].

Tabelle 5: Das Synset zur LU „buy“ in WordNet

Wortart	Tokens(s)	Erklärung	Beispiel
Substantiv	bargain, buy, steal	Ein vorteilhafter Kauf	She got a bargain at the auction
Verb	buy, purchase	erhalten durch den Kauf	The family purchased a new car
Verb	bribe, corrupt, buy	Bestechung	This judge can be bought
Verb	buy	Wahrnehmung	I can’t buy this story

Das nächste lexikalisch-semantisches Wortnetz ist GermaNet, die von der Struktur her dem WordNet entspricht. Im Kapitel 2.4.4 wird der Aufbau der GermaNet-Datenstruktur anhand von OWL-Modellen gezeigt. Die häufigsten und wichtigsten Konzepte des deutschen Grundwortschatzes sind in dieser Ressource enthalten. Wie in der obigen Tabelle zusehen ist, besteht WordNet aus Synsets und Lexical Units (buy, purchase), wobei zwei oder mehrere lexikalische Einheiten Synonyme sind, falls sie sich im selben Synset befinden. GermaNet wurde im gleichen Stil von WordNet entwickelt. Sie betrachtet diese zwei (Synset und LU) als lexikalische Objekte. Eine Beziehung (Relation) besteht sowohl zwischen diesen zwei Objekten als auch zwischen zwei gleichartigen Objekten der GermaNet-Datenstruktur (Synset-Synset oder LU-LU), die im [KLLS06, S.91.-92] als E.R.-Diagramm dargestellt wird. Eine Relation zwischen zwei Synsets heißt konzeptuelle Relation (CR) (z.B. Hyperonymie), wobei die LU-LU-Relation als lexikalisch-semantische Relation (LSR) genannt wird (z.B. Antonymie). Von der Art her kann sich Synset in Substantive, Verben, Adjektive sowie in Adverbien unterteilen. Man könnte das Synset aus der obigen Tabelle nämlich in Noun- bzw. VerbSynset aufteilen. Diese hierarchische Strukturierung (Synset → NounSynset | VerbSynset) kann man auch in LUs durchführen (LU → NounUnit | VerbUnit). Mit der Einschränkung der Synsets schränkt man auch dazugehörige LUs ein. Bei einer Einschränkung des obigen Synsets auf NounSynsets würden die VerbUnits (purchase, corrupt, bribe) ausfallen. Diese Einschränkung wird sich auch auf die OWL-Darstellung auswirken. Ein NounSynset würde gleichzeitig bedeuten, dass nur

#NounUnit‘s berücksichtigt werden kann und mit dem Tag <owl:Restriction> eingeschränkt werden muss (siehe ebenda). Für weitere Details über GermaNet-OWL-Modellierung siehe Kapitel 2.4.4.

Als letzte lexikalische Ressource, wird hier VerbNet vorgestellt, die zur Lösung des Problems der Prädikat-Argument-Strukturen (siehe Kapitel 2.4.1) eingesetzt werden kann. Eine VerbNet-Klasse beinhaltet sowohl syntaktische als auch semantische Informationen zu einem Verb. Sie enthält eine XML-Struktur und besitzt Mappings zu anderen lexikalischen Ressourcen wie WordNet oder FrameNet (siehe [Palm11]). Alle Verben im selben Frame besitzen gleiche syntaktische und semantische Eigenschaften. So besitzen das Wort „murder“ und alle anderen Verbmitglieder in Tabelle 6 die gleiche Semantik. Ein Verb kann aufgrund der Ambiguität in mehreren Klassen vorkommen, was dazu führt, dass Argumente verschiedene semantische Rollen annehmen können. Die Sätze “I opened the door “und “The key opened the door“ führen auf unterschiedliche semantische Argumente bezüglich des Verb „open“ hin. Im ersten Satz ist „I“ ein Agent, wobei im zweiten Satz „The key“ dem Instrument als Rolle entspricht. Weiterhin enthält eine VerbNet-Klasse folgende Informationen: Menge zugehöriger Mitgliedsverben, Sammlung von semantischen Rollen sowie syntaktischen Frames.

Tabelle 6: VerbNet-Klasse (entnommen aus [Scha10])

Klasse	**Murder-42.1**			
Verbmitglieder	Annihilate, assassinate, butcher, dispatch, eliminate, euthanize, execute, exterminate, immolate, liquidate, lynch, massacre, murder, off, slaughter, slay			
sem. Rollen	AGENT [+animate] PATIENT [+animate] INSTRUMENT			
Frames	Beschreibung	Beispiel	Syntax	Semantic
	NP V NP	Brutus murdered Julis	NP AGENT V NP PATIENT	Cause (AGENT, E) alive(start(E), PATIENT) not(alive(result(E), PATIENT)
	NP V NP PP.Instrument	Caesar killed Brutus with a knife	NP AGENT V NP PATIENT with NP INSTRUMENT	Cause(AGENT, E) alive(start(E), PATIENT) not(alive(result(E), PATIENT)) use(during(E), AGENT, INSTRUMENT)

2.2 Methoden der NLP

Die im [Mitk03] erwähnten Methoden der Cl werden hier aufgrund ihrer Vielzahl nicht alle durchgenommen. Davon werden drei Methoden exemplarisch behandelt, die im Laufe der Arbeit gebraucht werden.

2.2.1 POS-Tagging

Tagger bearbeitet einen mit Tokens gekennzeichneten Text, der nach einer Tokenisierung zustande gekommen ist. Der Tokenisierungsvorgang ist zuständig, Leerzeichen, Punkte oder auch Wörter zu erkennen, die die kleinsten Einheiten eines Textes bzw. Satzes bilden. Somit entstehen Tokens, die durch ein Tagger zur weiteren Verarbeitung zur Verfügung steht. Die meisten Tagger übernehmen die Aufgabe der Tokenisierung, die vor dem Tagging stattfindet. Beim eigentlichen Tagging-Vorgang, wofür auch der Tagger hauptsächlich zuständig ist, werden dann Wortarten zu den gebildeten Tokens zugeordnet. Bei der Bestimmung der Wortart werden lexikalische Ressourcen zur Hilfe genommen. Ein Lexikon wie GATE stellt eine Liste von Annotationen zur Verfügung [Scha10]. Vor der Annotation bzw. Tagging müssen zuerst die Grundformen der Tokens anhand einer morphologischen Analyse ermittelt werden, damit eine Wort-Wortart Zuordnung stattfinden kann.

2.2.2 Parsing

Parsing bestimmt die grammatikalische Struktur eines Satzes und stellt sie wie in Abbildung 1 anhand von Konstituentengrafen dar.

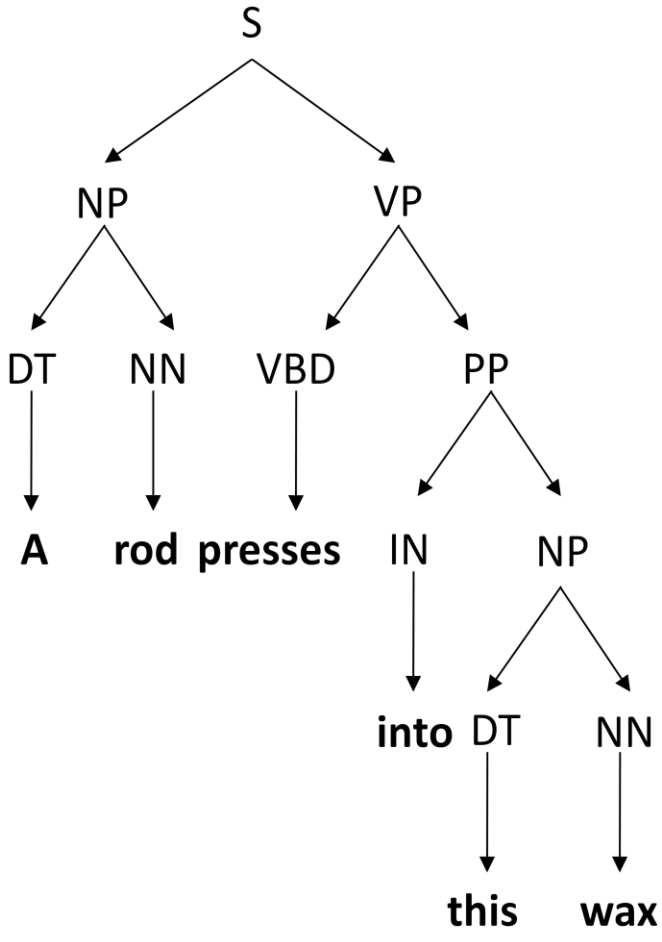

Abbildung 1: Stanford Parser – Konstituentenstruktur

Konstituenten sind Bausteine, die durch Zerstückelung von größeren Bausteinen (Sätzen) entstehen. Diese Strukturen können anhand eines Stanford-Parsers dargestellt werden. Ein Stanford Parser arbeitet statistisch und kontextfrei. Das Ergebnis ist ein Baum, der die syntaktische Struktur des Textes angibt. Diese Strukturen können für weitere semantische Analysen eingesetzt werden. Eine Definition hierzu lautet: "..., parsing, or, more formally, syntactic anlaysis, is the process of analyzing a text, made of a sequence of tokens (for example, words), to determine its grammatical structure with respect to a given (more or less) formal grammar" [Mitk03, Kapitel Parsing]. Eine vertiefte Behandlung des Themas Parsing ist hier nicht vorgesehen. Nichts desto trotz ist es für weitere Kapiteln hilfreich, Terme wie Konstituente, Nominal- bzw. Verbphrase, die aus der Abbildung 1 entnommen werden können, zu wissen.

2.2.3 Partial-Parsing

In den meisten Fällen einer Textanalyse ist es sinnvoll, nicht alle Wörter des Textes zu kommentieren. Daher lässt man bewusst einige Teile der Phrasen weg, deren Semantik in anderen Konstituenten des Satzes enthalten ist[Gelh10]. Hier muss klar gestellt werden, dass vom Text inhaltlich nichts verloren gehen darf. Die Auskommentierung von Textbereichen erleichtert letztendlich auch die Modellierung erheblich.

2.2.4 Ontologie

Nach einer Einführung in die Ontologie wird deren Gebrauch bei der Verarbeitung von natürlichen Sprachen vorgestellt.

Nach (Gruber,1995) gilt folgende Definition für die Ontologie: "[...] is an explicit specification of a shared conceptualization". Das Ziel in der Ontologie ist es also das implizite Wissen zu explizieren. Ein System kann mit dem impliziten Wissen wenig anfangen, wobei es für Menschen verständlich ist. Wenn ein Mensch weiß, dass der Zug um 7 Uhr abfährt (implizit), dann kann er selbst folgern, dass er nach diesem Zeitpunkt auf den nächsten Zug warten muss (explizit), obwohl es nirgendwo so steht. Ein Computer-System kann das also nicht beurteilen. Das Wissen muss aufbereitet (bzw. expliziert) werden, damit das System es verarbeiten kann. Ontologie ist eine Methode, die Wissen dem System zur Verfügung stellt. [Mitk03] kennzeichnet die Ontologie wie folgt: "[...] as an inventory of the objects, processes, etc. in a domain, [...]". Die Informatik benutzt diesen Begriff als eine Beschreibung einer bestimmten Domäne. Eine Domäne besteht aus nicht trivialen Verknüpfungen zwischen Objekten, deren Struktur weit über einer Hierarchie hinausgeht [Fürn10].

Ein Ontologie-Wissen hilft Texte zu bearbeiten. Für die Bearbeitung braucht man Information. Eine Information über ein Token (Wort) erhält man entweder aus einem Lexikon, in dem die Grundform und ihre Wortart enthalten sind, oder aus einer Wissensbasis (engl. knowledge base), indem Beziehungen zwischen Objekten zu finden sind. In letzterem Fall handelt es sich um eine formale Definition der Welterkenntnis und wird Synonym zu Ontologie gebraucht [Mitk03].

Nach [AEW08] dient die Darstellung des Wissens als eine Prozessoptimierung im Automotive Engineering Bereich. Auch in NLP benötigt man geeignete Ansätze zur strukturierten Speicherung und Darstellung von Zusammenhängen. OWL ist ein RDF-basierter Ansatz in der Ontologie, die für Wissensdarstellungszwecke dient. Andere Ansätze zur Strukturierung des Wissens sind Prädikat-Argument-Strukturen sowie Topic Maps, die in Kapitel 2.4 behandelt werden.

Ontologie sowie semantische Ressourcen (Datenbanken) werden für die Verarbeitung der natürlichen Sprache in den Anwendungen Information extraction (IE), Information retrieval (IR) [Mitk03, S.473 f.] sowie in Frage- und Antwortsysteme eingesetzt [UMA04, S.9].

IR-Information Retrieval- ist der Erhalt von im Speicher befindlichen Daten durch angereicherte Queries. Bei der Anreicherung werden normale User-Queries mithilfe von Synonymen und Homonymen umformuliert. Für solche Umformulierungen werden meistens Ontologien oder auch Word Netze integriert. Aufgrund der Ambiguität der Wörter erhält man leider mehr unpassende Ergebnisse (z.B. falsche Dokumente), als ursprünglich gewollt. Durch den Einsatz von Ontologie kann also das Ergebnis des Information Retrieval sich verschlechtern. Natürlich können Benutzereingriffe in solche Queries die Suchergebnisse positiv beeinflussen [Mitk03, S. 474].

IE-Information Extraction- dient zur Identifizierung von Entitäten und Erhalt von Eigenschaften dieser Entitäten. Zur Erkennung von Entitäten und Fakten werden semantische Netzwerke eingesetzt. Darüber hinaus wird mit IE versucht, Ereignisse im Text raus zu finden (siehe Information Extraktion). Diese Vorgänge entsprechen einem vordefinierten Template, deren Semantik auf Ontologie basieren kann [Mitk03, S. 474].

SUMO – Suggested Upper Merged Ontology- wurde als formale Ontologie entwickelt und ist frei erhältlich. Für SUMO gibt es den SUMO Browse- und Search-Tool, das die Mapping-Eigenschaft (lexikalische Ressource zu Ontologie) von SUMO realisieren kann und heißt SIGMA. Die große formale Ontologie SUMO wurde durch öffentlich verfügbare Ontologie-Inhalte wie communications, countries and regions, distributed computing, economy usw. zusammengebaut (vgl. [NiPe03, S.1], [SUMO11]). Beim Mapping vom WordNet zu (SUMO-) Ontology bietet SIGMA als Zielsprache neben der SUO-KIF Language auch provisorisch die OWL Language an. Durch dieses Tool erhält man also entweder SUO-KIF oder OWL als formale Sprache bzw. als Mapping-Sprache für die SUMO Ontology, wobei erstere die verlässlichere und OWL etwas verlustbehaftete davon ist. Bei der Übersetzung von WordNet zu OWL-Format steht nämlich im Kommentarteil zwischen <rdfs:comment> und </rdfs:comment> des RDF-Dokumentes folgendes: „A provisional and necessarily lossy translation to OWL. Please see […] for the original KIF, which is the authoritative source„ [Sumo11]. Der Grund für die Zuverlässigkeit des SUO-KIF hängt wahrscheinlich von der prädikatenlogischen Struktur des SUO-KIF ab. Die Prädikatenlogik ist ein Kompromiss für die Berechenbarkeit und die Ausdrucksstärke. Weitere Details über die Konvertierung von Wortnetzen in OWL, die nicht Entscheidbarkeit sowie hohe Ausdrucksfähigkeit einiger OWLs werden in Kapitel 2.4.4

behandelt. SUO-KIF wurde vom KIF abgeleitet (vgl. [Peas09, S.3], [Gene92]) und unterstützt die Definition von SUMO [NiPe01].

2.3 Anwendungen der NLP

In den vorherigen Kapiteln wurden aufeinander aufbauende Grundlagen und Methoden behandelt, die man hier mit Anwendungen ergänzen möchte. Anders gesagt kommt man auf diesem Weg vom theoretischen Teil zum Anwendungsteil.

Zu den Anwendungen in NLP gehören Information extraction, Information Retrieval oder auch Question & Answering, die man alle als SRL-Systeme kennzeichnet [Scha10]. Bei SRL werden Tokens mit semantischen Annotationen versehen und es steht für Semantic Role Labeling. Darüber hinaus wird in dieser Arbeit die Model extraction in die Gruppe der Anwendungen der NLP eingestuft, die die Entitäten mit ihren semantischen bzw. thematischen Rollen in einem Modell darstellt.

2.3.1 Information-Extraktion

Die IE ist die Gewinnung von wichtiger Information aus einem gegebenen Text. Eine der wichtigsten Informationen der NLP sind Namen, die mit einem NAME-Tagger identifiziert und klassifiziert werden können.

Ein Name-Tagger erkennt mithilfe eines Musters und einer Liste von Wörtern alle Substantive und stellt sie strukturiert dar. Als Ergebnis bekommt man eine SGML Mark-Up Struktur des Textes. Passende Typen zum Name sind dabei z.B. Menschenname, Organisation oder Ortsnamen.

Die Erkennung, dass z.B. XY eine Organisation ist, erfolgt mit dem Pattern, von denen einige in [Mitk03, S. 546 f.] vorgestellt wurden

capitalized-Word + 'Corp.'.

Einem Name-Tagger wird eine große Bedeutung beigemessen, da die meisten Texte viele Namen enthalten. Die Name-Identifikation und Klassifikation wird daher als eine wichtige Vorstufe in der Textanalyse angesehen. Da die meist relevanten Indizes Substantive sind [Mitk03, S. 549], werden sie in Dokumentenverwaltungssystemen eingesetzt. Vor allem spielt in dem Bereich in Name, Organisation oder Orte etc. eingeteilten Indizes eine wichtige Rolle.

2.3.2 Modell-Extraction

Nach einer semantischen Annotation mithilfe einer Annotationssprache (z.B. das Prototyp SALEMX) sind Grundlagen geschaffen, um ein Modell aus der semantischen Annotation extrahieren zu können. Bei SALEMX findet eine Transformation in einen Zwischengrafen (Omnigrafen) findet statt, bevor das Endmodell abgebildet werden kann. Ein Endmodell könnte dabei ein UML-Klassendiagramm sein. Das nächste Kapitel gibt eine Vorstellung, wie eine Modell-Extraktion in der Welt der natürlichen Sprache angewendet werden kann.

2.4 Modell-Extraktions-Ansätze in NLP

2.4.1 Prädikat-Argument-Struktur

Dieser Ansatz wird als einfachste Form der Wissenspräsentation angesehen und kann in der NLP eingesetzt werden. Die Prädikatenlogik der ersten Stufe wird in der Regel angenommen um werkzeugmäßig (z.B. mit Prolog) die (PAS) Prädikat-Argument-Struktur realisieren zu können. Erste Stufe weißt dabei einfache Sätze auf, die man in der Prädikatenlogik mathematisch darstellen kann. Die Darstellung des Satzes „Alle Wagen beinhalten eine Leistungssteuerungsoption“ in der Prädikatenlogik, ist folgendermaßen: $\forall x\ (W(X) \rightarrow b(X, L))$. Durch das Verb zuerst hervorgerufene Argument deutet auf ein Substantiv (Alle Wagen), das zweite Argument auf ein direktes Objekt (eine Leistungssteuerungsoption) [Gelh10, S.44]. Das einzige was hier zu merken ist, dass die Syntax auch berücksichtigt wurde.

Die Realisierung der PAS erfolgt mit Hilfe technischer Mitteln. Das Prolog Werkzeug diente hierfür. Anschließend wird versucht die PAS anhand eines Lexikons zu erkennen. Dafür nimmt [Scha10] das VerbNet, bekannt als größtes Englische Online Verb-Lexikon [Verb10]. Die hier erwähnte Realisierung gehört zu der syntaktischen Analyse wobei die Erkennung der PAS zur semantischen Analyse der NLP eingeordnet wird [Scha10]. Diese Strukturen können aus sogenannten Frames erkannt werden. Solch ein Frame von VerbNet wurde im Kapitel semantische Ressourcen vorgestellt. Basierend auf diesem Frame können die Erkennung der Struktur sowie die Annotation der semantischen Rollen zu den Satzgliedern erfolgen. Der folgende Satz ist nach folgenden Schritten semantisch annotiert: Erkennen der Prädikate (kill, follow) und Argumente (prep(killed, with)) mithilfe von syntaktischen Informationen, erkennen der Prädikat-Argument-Struktur, Matching zuVerbNet-Frames und Annotation der semantischen Rollen zur erkannten Struktur [Scha10].

Tabelle 7: Semantische Analyse eines unstrukturierten Satzes

Eingabe: unstruktu-rierter Satz	The masked criminal followed the man into the house an killed him with a knife
Ausgabe: annotierte Rollen im Text	[The masked criminal]Agent killed [him]Patient with [a knife]Instrument

2.4.2 Topic Maps (TM)

Topic Maps, deren grafische Struktur einer Landkarte entspricht, kann zu den semantischen Ansätzen der NLP zugeordnet werden. Sie sind auf SGML Basis eine ISO-Standard, was diesen Ansatz aufgrund seiner Eindeutigkeit relevant macht. Die Realisierung dieser Karten erfolgt in der Regel als XML-Datei.

Weitere semantische Modelle, die sich den TopicMaps bezüglich ihres hierarchischen Aufbaus ähneln, sind Taxonomie, Thesaurus und Ontologie. Taxonomie oder Thesaurus erlauben hierarchische Beziehungen zwischen Objekten. Thesaurus als mächtigeres davon erlaubt zwei feste Relationen. Diese sind Ähnlichkeits- und Synonymrelationen zwischen zwei Objekten. Nun sind mit TM mehr selbstdefinierbare Relationen zwischen den Knoten in der gesamten Hierarchie möglich. Ontologie wird nach [UMA04, S. 2] in der Evolution der semantischen Modelle als der mächtigste Ansatz zwischen diesen vier angesehen. Obwohl TM und Ontologie ein und dasselbe anbieten, bevorzugen [UMA04, S. 10] die Ontologie, da sie zusätzlich zu dem TM noch integrationsfähiger sind, weil sie als Middleware zur Verfügung gestellt werden können. Für eine detaillierte Erklärung über TM können diese Literaturen herangezogen werden ([Gelh01], [Gelh10]).

TM ähneln sich den HG (Hypergrafen). Beide erlauben n-äre und direkte Beziehungen (Verknüpfungen) zwischen Kopf- und rollenbehafteten Nicht-Kopftöchtern (siehe unten). Dabei ist bemerkenswert, dass bei beiden Grafen eine Kante mehrere Enden erlaubt. Aufgrund dieser Enden sind TM gegenüber HG flexibler. TM sind bezüglich ihrer abstrakten Konzepte bzw. ihrer mächtigeren Struktur gegenüber PAS oder OWL offensichtlich die bessere Abbildungsmöglichkeit für die natürliche Sprache, obwohl z.B. TM und PAS theoretisch gleich mächtig seien (siehe Ebenda). Nichtsdestotrotz ist OG, wie im nächsten Kapitel zu sehen ist, die bessere Variante, da sie mehr wie die hier genannten Konzepte leisten kann. Sie wurde auch im Rahmen der Arbeit von [Gelh10] entwickelt. Subject Identifier (links) ist die einzige Kennzeichnung(ein Link), wo diese TM auch vorkommt.

Kopf- und Nicht-Kopftöchter bilden zusammen eine Konstituente. Wie schon in vorherigem Kapitel erwähnt, sind Konstituenten kleinere Einheiten (Teilkonstituenten), die zusammen eine größere Einheit (Konstituente) einer Sprache bilden. Die kleinste Teilkonstituente ist ein einziges Morph (Wort). Der Satz „ein schönes Haus“ ist eine Konstituente, die aus kleineren Konstituenten besteht, wovon nur das Wort „Haus“ die Kopftochter (da sie ein Spektrum aufweist) und die anderen, die beschreibenden Tochter sind, weshalb auch die letzteren als Nicht-Kopftochter bezeichnet werden [Gelh10].

2.4.3 Omnigrafen (OG)

Omnigrafen bestehen wie andere Grafen (z.B. TM) aus Knoten und Kanten. Da aber TM (=Vorstufe für Omnigrafen) oder auch andere Grafen wie Hypergrafen (=ähnliche wie Omnigrafen) nicht in der Lage sind zweierlei Assoziationen miteinander zu verknüpfen (also Relationen der Höheren Ordnung), wird der OG als Zwischengraf vor dem UML-Modell bestimmt. Die Knoten (Omniknoten) dieses Grafen bestehen hauptsächlich aus den offenen Wortklassen, wobei die geschlossenen Wortklassen für die Verbindung dieser Knoten zuständig sind (Omnikante). Strukturkonzepte der TM gelten auch bei den OG (siehe Kapitel 3.3.1). Dadurch besteht ein OG auch aus einer seriellen XML-Struktur. Gleichzeitig dient OG als Grundstruktur für SENSE (The Software Engineer’s Natural-language Semantics Encoding), die die Semantik-Kodierung in SALEMX realisiert. (siehe Ebenda).

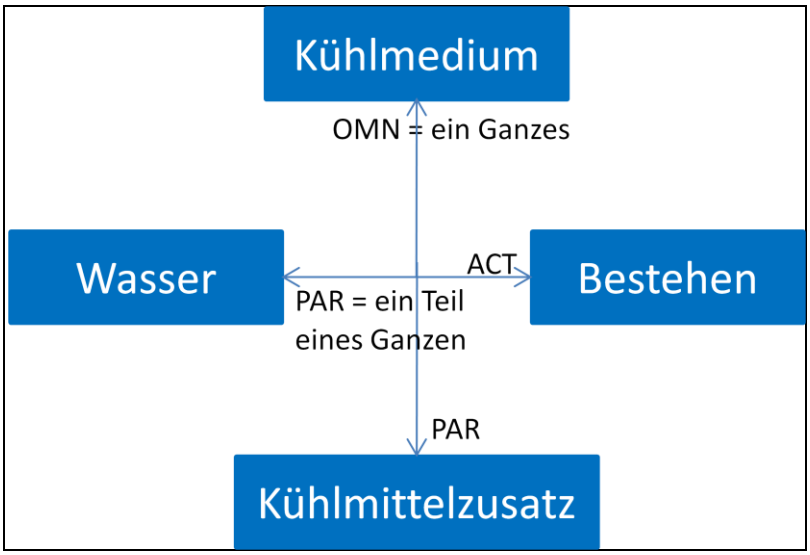

Abbildung 2: N-äre Relation eines OG (vgl. [Wies11])

Eine formale Definition für OG wurde in [Gelh10] angegeben. Der folgende Satz wird nun schrittweise in einem OG modelliert: Ein Kühlmedium besteht aus Wasser und einem Kühlmittelzusatz. Zuerst werden die Knoten (Kühlmedium, Wasser, Kühlmittelzusatz, besteht), die Kante (Assoziation) und die Rollen (PAR, OMN, ACT, PAR) des OG bestimmt. Als nächstes müssen aufgrund der Definition des OG die Tt (Tentakel) notiert werden. Tt sind nach [Gelh10] die Endstellen der Kanten anders gesagt die Anschlussstellen der Kanten zum jeweiligen Knoten. Somit sind Tt im OG entweder eine Verbindung zwischen Kantenende und Knoten oder wenn eine Kante zu Kante-Beziehung (was in OG ja zulässig ist) besteht, eine Verbindung zwischen Kantenende und einer anderen Kante (siehe Abbildung 3). Tentakel ist die Stelle im Grafen, wo sich die Pfeilspitzen befinden. Die einzige Omnikante namens „Assoziation“ erlaubt in diesem Beispiel vier Enden. Da hier vier Kantenenden existieren, so geht man auch von vier Tt aus, die jeweils in drei Funktionen als Eingabeparameter ausgewertet werden. Angewendet auf die rol, src und tar –Funktionen erhält man dann für Tt_1 (=Bestehen) folgende Werte: rol(Tt_1)=ACT für die Rolle der Tt_1, src(Tt_1)=Assoziation für den Ursprung und schließlich die Anschlussstelle für Tt_1 mit tar(Tt_1)=Bestehen.

In der unteren Abbildung besteht der OG aus sechs Tt, was durch die Anzahl der Pfeilspitzen sich leicht rechnen lässt. Dabei können wie im obigen Graf einzelne Objekte (z.B. Knoten, Kanten, Tentakel) ermittelt werden. Die Zwischen-Relation (mittig), die die zwei Relationen miteinander verbindet ist eine XOR-Relation. Hier gibt sie an, dass der Motor entweder im Voll- oder in Teillast sich befinden kann.

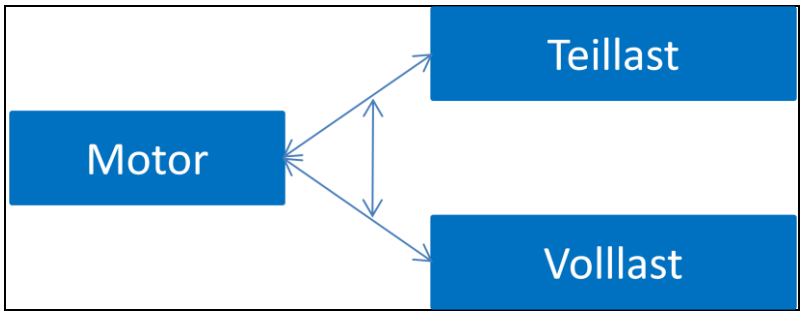

Abbildung 3: N-äre Relation eines OG mit höherer Ordnung (vgl. [Wies11])

Da in den Arbeiten von ([Gelh10] [Den07]) bezüglich des Omnigrafen (Supergrafen bei [Den07]) vieles detailliert erklärt worden ist, besteht hier kein Bedarf zur Behandlung dieses Themas. Interessierte werden auf diese zwei Werke verwiesen.

2.4.4 Web Ontologie Language (OWL)

Das letzte Wissenspräsentationsmodell OWL, das hier vorgestellt wird, dient in einer Domäne die Objekte und die komplexen Relationen zwischen diesen korrekt zu formalisieren. Dabei werden Klassen und Eigenschaften (Properties) verwendet, die später mit Beispielen oder im Anhang näher erläutert werden. Das Ziel von OWL und seinen Vorgänger XML, RDF(S), OIL und DAML ist es, eine Möglichkeit zu schaffen Dokumente bzw. Wissen strukturiert abzuspeichern. XML bietet zwar die Syntax solcher strukturierten Dokumente an, aber eine semantische Interpretation dieser Dokumente ist nur durch Menschen realisierbar. RDF (Ressource Description Framework) dient als Datenmodell für Objekte und deren Relationen und bietet eine recht einfache Semantik. Mithilfe der XML-Syntax beschreibt RDF die Objekte und Relationen im Datenmodell. Die Idee der Ontology der W3 Consortium stammt aus der Zeit, in der RDF entwickelt wurde. Danach wurde RDFS entwickelt, um RDF Eigenschaften und Klassen zu beschreiben. Neben dieser Beschreibungsfähigkeit war durch RDFS auch die Generalisierung möglich. RDFS ist aber für die Implementierung des Semantic Webs nicht geeignet da sie einige Mängel aufweist. Es fehlen z.B. wichtige Relationen zwischen den Klassen wie Äquivalenz oder Kardinalität usw. und die Ausdrucksfähigkeit ist unangemessen. RDFS leidet aufgrund der Abhängigkeit von domänenspezifischen Details. Diese Probleme wurden versucht durch OIL und DAML zu lösen, die auf RDFS aufbauten. Später wurden diese zwei Sprachen in einem Standardisierungs-Prozess von W3C kombiniert und anschließend durch OWL verbessert [ArZh04]. Eine Übersicht über die Architektur des Semantic-Web, die die hier erwähnten Sprachen beinhaltet, befindet sich in [Bern00].

Technisch gesehen beinhaltet OWL alle RDF Eigenschaften und nutzt deren Syntax. Die einfache (Tripple-) Semantik (bzw. zweistellige Prädikate in Prädikat-Argument-Struktur), wie in Abbildung 4 mit Objekt (Startknoten), Attribut (Kantenbezeichnung) und Wert (Endknoten) zu sehen ist, erbt OWL von RDF. Die Darstellung der GermaNet-DB erfolgte durch OWL. Falls „isHyponymOf“ dieses Mal als Startknoten einen weiteren Knoten besitzt, würde die Grafik sich nach rechts erweitern. Dabei ist der Startknoten dann das Objekt, und der Endknoten

wiederum Wert des Objektes. OWL erweitert all diese Eigenschaften mit eigenen Features, die sie dann bezüglich der semantischen Interpretationsfähigkeit noch mächtiger macht. Zwar ist OWL seriell aufgebaut, kann aber gleichzeitig visuell als ein gerichtetes Graf dargestellt werden. Im SemanticWeb setzt sich OWL gegenüber den anderen XML, RDF(S), aufgrund der Fähigkeit maschinenlesbaren bzw. interpretierbaren Kontext bereitzustellen, besser durch. (vgl. [ArZh04], [Gelh10]). Inwieweit sich OWL aber für NLP eignet, oder ob sie sich gegenüber andere (z.B. Omnigrafen) in diesem Kapitel erwähnte Wissenspräsentationsmodelle in Bezug auf NLP durchsetzt, wird sich noch herausstellen.

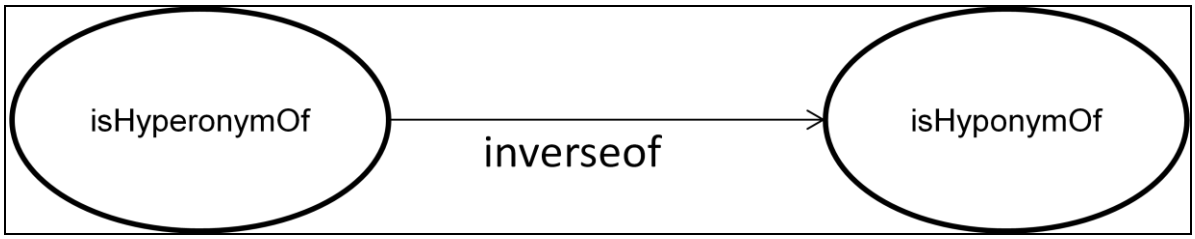

Abbildung 4: RDF-Trippel als grundlegende OWL-Eigenschaft (vgl. [Fürn10], S.38)

OWLs werden bezüglich des Umfangs untereinander unterschieden. Es gibt drei Arten, die in der Literatur erwähnt werden. Diese sind OWL Lite, OWL DL (Description Logic) und OWL Full. Je nach Anwendungsfall wird das eine oder das andere für den OWL-Entwickler relevant sein. Hier ist immer ein Trade-Off zwischen Ausdrucksstärke und Reasoning-Fähigkeit – eine Fähigkeit, die auf logisches Denken basiert - zu erkennen. Die Komptabilität zu RDF und die Berechenbarkeit solcher Sprachen sind bei der Wahl auch entscheidend. Die Tabelle 9 listet die OWL Arten sowie bestimmte Merkmale auf. Detaillierte Erklärungen gibt es in der Literatur [ArZh04].

Tabelle 8: OWL Arten und Mächtigkeit

OWL-Arten	Ausdrucksstärke	Vernunftbegabung (Reasoning)	Berechenbar/ entscheidbar	Kompatibel zu RDF(S)
OWL Full	sehr hoch	nicht möglich	Nein	Ja
OWL DL	eingeschränkt	effizient	Ja	Nein
OWL Lite	eingeschränkt	sehr effizient	Ja	Nein

Durch das Mapping von lexikalischen Ressourcen zu OWL wird versucht, den reichlichen Informationsgehalt von Lexikons mit der hohen Ausdrucksstärke von OWL (im Falle von OWL Full sehr hoch) zu verknüpfen um damit beim NLP effizienter sowie formaler vorzugehen. Die OWL Version von Princeton WordNet erlaubt es WordNet in OWL zu konvertieren. Es existiert

in [W3C04] ein Projekt namens „Wordnet in RDFS and OWL“, die durch Wordnet Task Force entwickelt wird.

Die im Kapitel 2.1.3 erwähnte Ressource GermaNet wurde prototypisch mit 37 (v. insg. 53 Tausend) Synsets und 83 (v. insg. 76 Tausend) LexicalUnits sowie mit den Relationen zwischen ihnen als Ontology in OWL kodiert. Von der Art der Ontology wurde in GermaNet OWL DL genommen, da die meisten Systementwickler diese Subsprache für ihre Systeme relevanter halten. Die Kodierung erfolgte mit dem Ontologie-Editor Protégé, wobei als Ausgangsformat entweder die XML- oder die Prolog-Version von GermaNet dient. Bei diesem Vorgang hat man als aller erstes die Konzepte des GermaNets nämlich Synset und LU als Klassen definiert (owl:class). Die Relationen zwischen den Klassen sowie die Eigenschaften der Klassen werden als Properties kodiert (Object- und DatatypeProperty). ObjectProperties sind solche Properties, die die Verknüpfung zwischen den Klassen Synsets und LUs realisieren. Für die Relation dieser Klassen bzw. Objekte sorgt ein sog. Object Property. ObjectProperties eignen sich also für die Darstellung der Relation zwischen den Klassen. Kombinationsmöglichkeiten solcher Verbindungen für die genannten Klassen sind: Synset-LexicalUnit, Synset-Synset oder LexicalUnit-LexicalUnit. Properties können mit dem Konstrukt <rdfs:subPropertyOf> genauso wie bei den Klassen <rdfs:subClassOf> hierarchisch angeordnet werden. Neben den Properties wie hasMember (siehe Abbildung 5), memberof gibt es in GermaNet noch andere Properties, die eine Relation zwischen zwei gleichen Klassen (z.B. LexicalUnit-LexicalUnit) sind. Diese heißen im Falle von zwei Synsets konzeptuelle Relation (CR) und im Falle von zwei LexicalUnits lexikalisch-semantische Relation (LSR). Alle diese Properties werden als TopLevel Properties genannt. Daneben gibt es Subproperties, die von CR oder LSR abgeleitet werden können (Abbildung 7 und 8). Eine Subklasse NounSynset ist in Abbildung 7 gegeben. Eine Subklasse für NounUnit als Teilmenge einer LexicalUnit kann ähnlich wie in Abbildung 7 zusammengebaut werden. Die untere Tabelle gibt eine Gesamtübersicht von all den hier erwähnten und für die OWL-Struktur relevanten Konzepten des Germanets.

Tabelle 9: Charakteristika der Object Properties (vgl. [KLLS06, S.93])

Property	Domain	Range	Charakteristika	Inverse Property	Lokale Restriktion
hasMember(Top Level)	*Synset*	*Lexical Unit*	*Invers-funktional*	*memberOf*	*wortart-bezogen*
Memberof (Top Level)	*Lexical Unit*	*Synset*	*funktional*	*hasMember*	*wortart-bezogen*
isHyperonymOf (Subproperty von CR)	*Synset*	*Synset*	*transitiv*	*isHyponymOf*	*wortart-bezogen*
hasAntonym (Suproperty von LSR)	*Lexical Unit*	*Lexical Unit*	*symmetrisch*	*hasAntonym*	*wortart-bezogen*

Die Kodierung der Datatyp Properties erfolgt durch Zuordnung von Wertebereichen zu den OWL Klassen. Mithilfe von XML-Schema-Datentyps werden Strings, Booleans oder andere linguistische Wertebereiche wie Nouns, Verb, Adj. Adv. etc. zugewiesen (siehe Tabelle 11).

Tabelle 10: Charakteristika des Datatyp Property POS (vgl. [KLLS06, S.94])

Property	Domain	Range
POS	*Synset*	*„N“|„V“|„A“|„ADV“*

Die durchgeführten Hervorhebungen in den kommenden OWL Strukturen haben mit der OWL-Syntax erstmals nichts zu tun. Sie wurden gemacht, damit Klassen und Properties vom Leser direkt identifiziert werden können. Dabei sind die klassen- bzw. relation-artige Bezeichnungen mit rot bzw. grün gekennzeichnet.

```
1  <owl:Class rdf:ID=“NounSynset“ >
2    <rdfs:subClassOf >
3       <owl:Restriction >
4          <owl:allValuesFrom>
5             <owl:Class rdf:about=”#NounUnit” / >
6          </owl:allValuesFrom>
7          <owl:onProperty>
8             <owl:InverseFunctionalProperty rdf:ID=”hasMember” / >
9          </owl:onProperty>
10       </owl:Restriction >
11   </rdfs:subClassOf >
12 </owl:Class>
```

Abbildung 5: Sub-Klasse NounSynset Teilmenge von Synset (vgl. [KLLS06, S.92])

Das Hyperonym ist das Oberwort für ein betrachtetes Wort. Der spiegelbildliche Begriff hierzu ist das Hyponym. [Wiki11]. Man könnte also sagen, „A isHyperonymOf C“, dann gilt andersrum auch „C isHyponymOf A“. Angenommen „A isHyperonymOf B“ gilt und „B isHyperonymOf C“ gilt, so gilt auch aufgrund der Transitivität „A isHyperonymOf C“.

```
1 <owl:TransitiveProperty rdf:about=”#isHyperonymOf” >
2   <rdfs:subPropertyOf rdf:resource=”#conceptualRelation” / >
3      <rdfs:domain rdf:resource=#Synset” / >
4      <rdf:type rdf:resource=”http://www.w3.org/2002/07/owl#ObjectProperty” / >
5      <owl:inverseof>
6         <owl:TransitiveProperty rdf:about=”isHyponymOf” / >
7      </owl:inverseof>
8 </owl:TransitiveProperty>
```

Abbildung 6: Subproperty „isHyperonymOf“ als Relation zwischen Synset Klassen (vgl. [KLLS06, S.92])

```
1 <owl:SymmetricProperty rdf:about=”#hasAntonym” >
2   <rdfs:subPropertyOf rdf:resource=”#lexical-semantic Relation” / >
3      <rdfs:domain rdf:resource=#LexicalUnit” / >
4      <rdf:type rdf:resource=”http://www.w3.org/2002/07/owl#ObjectProperty” / >
5      <owl:inverseof>
6        <owl:SymmetricProperty rdf:about=”hasAntonym” / >
7       </owl:inverseof>
8 </owl:TransitiveProperty>
```

Abbildung 7: Subproperty "hasAntonym" als Relation zwischen LU. Klassen

Eine weitere Property ist das Antonym (Abbildung 7) mit der Bezeichnung „hasAntonym“. Hierbei handelt es sich um gegensätzliche Wörter, die ein gegebenes Wort enthält. Die symmetrische Property zu „hasAntonym“ ist daher dasselbe (Zeile 6).

3 Zielverwandte Arbeiten und Abgrenzung

3.1 Arbeiten mit gleichem Ziel der Modellextraktion

Im Bereich der MX gibt es zahlreiche Herangehensweisen, die entweder die direkte Abbildung des Modells in eine Programmiersprache ermöglichen oder zuerst ein Zwischenmodell erzeugen und anschließend den Code generieren. In der Letzteren, die in dem Bereich etwas besser erscheint, muss eine Grafersetzung oder Graftransformation mit Werkzeugen wie dem GrGen durchgeführt werden. Die Wartungsarbeit bei Anpassungen ist ein Grund, warum dieses Verfahren nicht genommen wird (vgl. [Den07, S.135 f.] und [Gelh10]). Die nächsten zwei Kapiteln und die untere Tabelle beschreiben genau die obigen zwei Vorgehensweisen für die MX in der NLP.

Tabelle 11: Zielverwandte Arbeiten ([Gelh10])

Publikation	Zwischenpräsentation	Ausgabe/Zielmodell	Sprachumfang
[Abb83]	nicht vorhanden	Ada-Programm	eingeschränkt
[Gelh10]	SENSE	UML-Klassen- und Aktivitätsdiagramme	nicht eingeschränkt

Die obige Tabelle stammt aus einer viel größeren Tabelle, die ähnliche Arbeiten in diesem Bereich enthält. Neben Grafen wie z.B. UML dienen auch Programmiersprachen als Endmodelle. Außerdem gibt es in einer anderen Arbeit [Den07, S. 46] ein Vergleich zwischen mehreren Werkzeugen, in dem auch Prolog und OWL als Werkzeuge verglichen werden, wobei man mit OWL zwar Wissen präsentieren aber nicht weiterverarbeiten kann und somit ungeeignet zur Graftransformation ist. Ein Startgraf ist durch OWL mit Klassen (für Knoten) und Relationen (für Kanten) darstellbar [Den07, S. 26]. Wie bei vielen Wissensdarstellungsmodellen besteht auch bei OWL das Problem, eine komplexere (Kante-Kante) Beziehung nicht darstellen zu können. Nach dem jetzigen Stand kann dies nur ein OG mit seinen Superkanten, was im Rahmen der Arbeit von [Gelh10] entwickelt wurde, realisieren.

3.2 Extraktion eines Modells in einer Programmiersprache

Zu den frühesten Arbeiten im Bereich von MX aus unstrukturierten Texten zählt das Werk [Abbo83]. Hier geht es vor allem darum, dass die Bezeichner einer Programmiersprache, egal wie komplex ein Algorithmus auch ist, präzise und verständlich ausgewählt werden sollen. Diese Idee wurde auch dort bei der MX in die Sprache Ada verwirklicht. Hierbei wurde geachtet, dass die Bezeichner für Datentypen, Variablen oder auch für Operatoren entsprechend ihr Vorkommen im Text eingesetzt worden sind. Somit entstand vom Text zum Ada-Programm

mit wenigen Ausnahmen eine 1:1 Abbildung. Nach einer mühsamen Formalisierungsstrategie wurde aus einem nicht formalen Text ein komplettes Ada Packet modelliert. Dabei ist zu beachten, dass es sich um einen vom Autor vorbereiteten eigenen Text handelt, der bewusst algorithmische Beschreibungen enthält.

3.3 Extraktion eines Modells durch Grafersetzung

3.3.1 SENSE (Software Engineer's Natural-language Semantics Encoding) und die Behandlung der Semantik von Wortklassen

SENSE ist eine Kodierungsmöglichkeit für die Semantik der natürlichen Sprache. Nun werden diese Möglichkeiten, beginnend mit den kleinsten Bausteinen nämlich, mit Wortklassen der deutschen Grammatik betrachtet. Zu den Wortklassen, die auch in SENSE teilweise kodiert wurden, gehören offene sowie geschlossene Wortklassen. In dem Blickwinkel eines Programmierers entsprechen die offenen Wortklassen den Bezeichnern bzw. Variablen und geschlossene Wortklassen den Schlüsselwörtern (Key-Wörtern) einer Programmiersprache. Diese Unterscheidung von Wortklassen in SENSE hat einen Grund, der beim Aufbau des Omnigrafen eine Rolle spielt. Der Omnigraf und deren Arbeitsweise wurden in Kapitel 2.4.3 bei Wissenspräsentationsmodellen behandelt. Eine grobe Übersicht der offenen Wortklassenarten mit Bezug auf den Begriff *Spektrum* können aus der Tabelle 13 entnommen werden. Eine ausführliche Erklärung über Wortklassen sowie über Semantik von komplexeren Strukturen von SENSE gibt es in [Gelh10, S. 89-159]. Die geschlossenen Wortklassen werden hier nicht behandelt. Diese sind vor allem Artikel (ein, einer, eines), Präposition (in, auf, an), Pronomen (meiner, deiner,…) und die Konjunktionen (jedoch, sondern, falls).

Ein Spektrum, ist ein wichtiger Begriff, der hier und im nächsten Kapitel gebraucht wird und dient dazu, Bezeichnern bzw. offenen Wortklassen ihre Rolle(n) im gegebenen Kontext zuzuweisen. Dabei ist zu beachten, dass einem Bezeichner mehrere Rollen zugeordnet werden können. Im Satz „Peter schenkt Anna Blumen" wird Peter zwei Rollen annehmen, nämlich „donor; DON" (der Gebende) und „agens; AG" (der Handelnde).

Angenommen, die semantisch sinntragenden Symbole (geschlossene Wortklassen) von SENSE, seien endlich. Dann kann ein Programm $\mathcal{P}$ diese Grundstrukturen erkennen und verarbeiten. Der folgende Satz „Die Schlüsseln sind im Haus" verarbeitet $\mathcal{P}$ in dem es die Schlüsseln und das Haus als Objekte und die in SENSE als Präposition kodierte „im" als eine geschlossene Wortklasse erkennt. Das Werkzeug SALEMX, das in Kapitel 3.3.3 vorgestellt wird, ist ein Prototyp für solch ein $\mathcal{P}$.

[Gelh10] nimmt als Basis den narrativen Stil der deutschen Sprache, in dem nur eine Zeitform (Tempus) nämlich Präsens oder Präteritum zulässig sind (keine Behandlung in SENSE; siehe Tabelle 13) und die dritte Person. Dieser Stil einer Sprache reicht auch völlig aus um Anforderungsbeschreibungen, was auch Ziel dieser Arbeit ist, zu modellieren.

Tabelle 12: Semantik der offenen Wortklassen und deren Kodierung in SENSE

<table>
<tr><td>Nomen, Namen:

(Modellierung als Knoten, da sie immer ein Spektrum besitzen)</td><td colspan="3">Länge, Unterschied, Meter, Watt, Gramm, Jahr, Stunde, Mai, Steuergerät, Tür, Straße, ESP, ABS, Dach</td></tr>
<tr><td rowspan="4">Verben:

(Modellierung als Knoten oder als Relation)</td><td>Vollverben: gehen, warten, gewinnen, liegen …</td><td>Modalverb: können, dürfen, müssen, sollen …

(Ausnahme: Modellierung als Relation in Omnigrafen)</td><td>Hilfsverb: haben, werden, sein …

(Semantik im Vollverb schon kodiert -> keine Kodierung in SENSE)</td></tr>
<tr><td>Person, Numerus:
Singular, Plural

(richten sich nach dem Substantiv)</td><td>Tempus (Zeitform):

(<u>Verzicht</u> auf Kodierung unterschiedlicher Zeitformen wg. einheitlicher Zeit)</td><td>Indikativer Modus: Auto fährt entlang der Straße

(entspricht dem, was tatsächlich geschieht</td></tr>
<tr><td colspan="3">Haben: Keine Sonderbehandlung in SENSE. Beispiel: Peter hat Auto; Modellierung durch zwei Konstituenten (Peter, Auto) + jeweilige Relation</td></tr>
<tr><td colspan="3">Sein: Keine Sonderbehandlung. Beispiel: Peter ist ein Arbeiter;

1. Lösung des Problems durch is-a-Beziehung -> nicht optimal

2. Lösung des Problems durch die SENSE-Rollen FIN und FIC (siehe Anhang)</td></tr>
<tr><td>Adjektive (dienen zur Modifikation, Modellierung in SENSE-Graf als Knoten vorgesehen, da sie ein Spektrum aufweisen)</td><td>Vergleichsformen: groß, größer, am größten, neueste …</td><td>Multiplizitäten: eins, zweiter, ein Drittel, vierfach oder fünferlei

(Ordnungszahlen, Bruchzahlen, Vervielfältigungszahlwörter, Gattungszahlwörter)</td><td>Quantoren: viel, wenig, zahllos, gering, alle, jeder, mehrere, beide, irgendein, manche

(als Begleiter für Nomen)</td></tr>
</table>

3.3.2 SALE-Syntax

Als Grundlage für SALE dient die SENSE, die im vorherigen Kapitel schon behandelt wurde. SALE ist eine Annotationssprache für linguistisch-semantische Information. Die SALE-Syntax ähnelt sich dem BNF aus der Informatik-Grundvorlesung. Diese Syntax wird im Kapitel 4 zusammen mit dem Mapping-Konzept behandelt und ist zudem im Anhang zu finden.

3.3.3 SALEMX: ein Prototyp für Programm $\mathcal{P}$

Das Gesamtsystem, das mehrere Komponenten für NLP und Model-Extraction enthält und das Prototyp zum vorher erwähnten (in 3.3.1) Programm $\mathcal{P}$ darstellt, heißt SALEMX. Die Erweiterung MX ist ein Akronym für die Modell-Extraktion. Die Komponenten von SALEMX-Werkzeug sind: SALE-Übersetzer + GRGEN.NET + Grafmodell + Regeln. Eine Darstellung des Werkzeugs SALE-MX ist in der unteren Abbildung angegeben. Die Prozessschritte des **M**odell-**E**xtractor wurden zwar in [Gelh10] oder in [Den07] ausführlich behandelt, wird hier aber nochmals mit den wichtigsten Bauelementen in Kurzversion und mit einigen Ergänzungen, bezüglich des leichteren Verständnisses über das Gesamtsystem, erläutert. Im ersten Schritt wird ein einfacher (fremder) Text mit der SALE-Syntax (manuell) annotiert. Die einfachste Einheit der Annotationsregeln für SALE bestehen aus einer Menge von Spektren und dazu gehörigen Rollen, so dass die thematischen Rollen wie AG, DON, PAT usw. im zweiten Rechteck zustande kommen.

Nach dem 2. Schritt entsteht der Omnigraf mit seinen Superkanten. Es entsteht also ein Graf, in dem Kanten auf Kanten verweisen können. Diese Art von Relationen sollen nun im 3. Schritt durch entsprechende Grafmodelle für UML sowie durch Transformationsregeln eliminiert werden. Eine komplexe Relation wird durch einen Knoten ersetzt. Schließlich erhält man ein Grafmodell, das nur noch aus normalen Relationen nämlich „Klasse-Assoziation-Klasse“, besteht, wobei in UML n-äre Relationen selten vorkommen aber zulässig sind.

Im Verlauf dieser Arbeit stand das GRGNET-System immer mehr im Vordergrund. Dies wollte man aber irgendwie umgehen. Neue Rollen für SALEMX würden nämlich bedeuten, dass neue Beziehungen zwischen den Knoten entstehen. Dies erfordert aber eine aufwändige Anpassung der Regeln im Grafersetzungssystem. Durch Absprache mit den Entwicklern des Projekts wurde anstatt dem GRGN.Net das EMF Projekt ausgewählt, und mit dem SALE-Compiler kombiniert. Der SALE-Text konnte nun durch den SALE-Compiler in EMF konforme Strukturen (sogenannte XMIs) übersetzt werden. Weitere Bemerkungen siehe Kapiteln 4 und 5.

Die Arbeit von [Land10] enthält die Entwicklung des AutoAnnotator (AA), das ein Plug-in zum SALEMX ist. Dieses probiert automatisch Rollen zu vergeben. Mithilfe von Benutzerinteraktion wird dieser Prozess optimiert. Die manuelle Rollenvergabe ist in der Abbildung durch den Schritt 1 zu erkennen. Die Aufgabe von AA ist es diesen Schritt zu automatisieren. Der erste Schritt von AA beinhaltet mehrere Vorgänge wie Tokenisierung, Morphogical Processing, POS Tagger etc. Dieser wird in [SNL01] als ein NLP-Pipeline bezeichnet. Nach der NLP-Analyse erfolgt die Unterstützung durch Ontologie-Konzepte [Land10]. Schließlich erhält man einen annotierten Text (bzw. SALE-Sprachmodell).

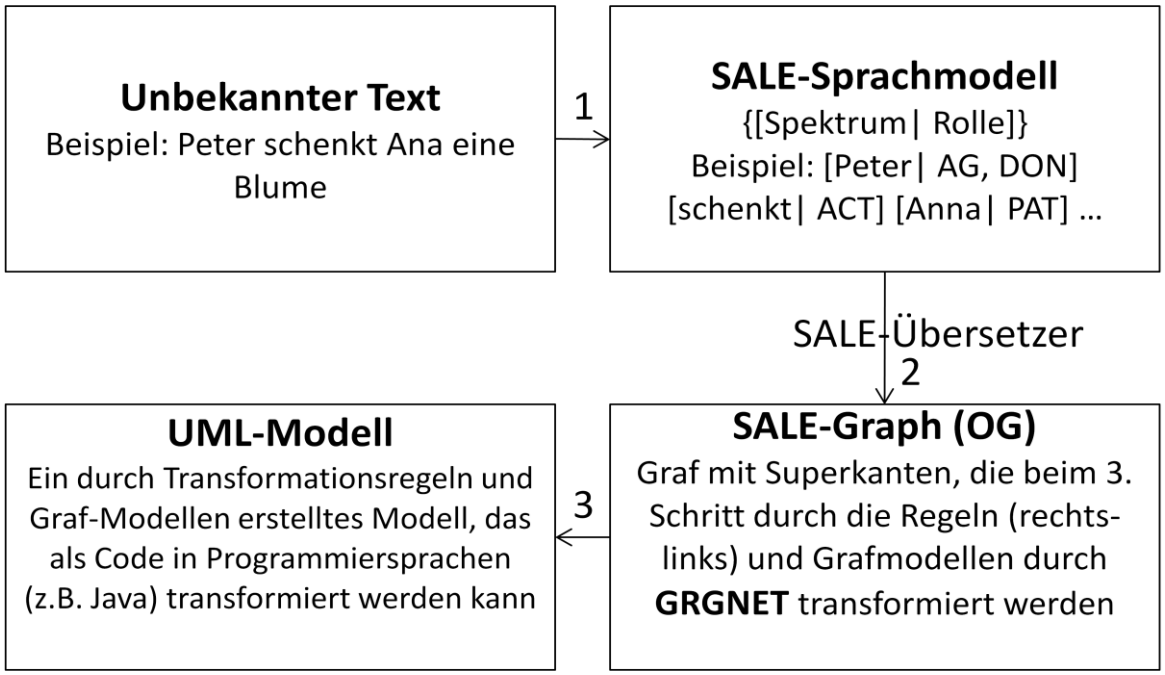

Abbildung 8: Prozessübersicht für SALE-MX (vgl. [Gelh10], [Den07])

3.4 WRSPM-Methode

WRSPM (World Requirements Specification Program Machine) ist ein Referenz-Modell für RE, das mit formalen Modellen arbeitet. Als wichtige Eigenschaft zählt in WRSPM die Unterscheidung von Phänomenen (Zustandsraum und -transition) und Artefakten (Einschränkung dieser Phänomene), die jeweils sowohl für die Umgebung als auch für das System bestimmt sind. Die Unterteilung dient z.B. für die Vermeidung von schwer erkennbaren Problemen, die in den frühen Phasen von großen Projekten mit mehreren Sub-Systemen auftreten können. WRSPM ist ein Akronym, das den fünf Artefakten (World, Requirement, Specification, Program, Machine) entspricht. Diese Artefakte werden durch die vier Phänomene (e_h, e_v, s_v, s_h) ausgedrückt bzw. Artefakte schränken diese Phänomene ein. So drücken bestimmte Phänomene der Environment (e_v) und des Systems (s_v) das Artefakt S (Specification), die der Verbindung zwischen der Welt und dem System entspricht, aus. Analog können (e_h, e_v) zu W und R sowie (s_v, s_h) zu P und M zugeordnet werden. Das Phänomen e_v ist vom System und s_v von der Umgebung aus sichtbar (v=visible). Die anderen zwei Phänomene (e_h und s_h), die ein $_h$ enthalten, sind für das System oder für die Umgebung nicht sichtbar (h=hidden). Neben der Sichtbarkeit gibt es die Kontrollierbarkeit als Eigenschaft. Einige Phänomene sind also vom System (s) und andere von der Umgebung (e) überwachbar. Das Artefakt „W“ schränkt die Welt in eine Miniwelt ein, woraus ein Domainmodell entsteht. Das „R“ schränkt dieses Modell in Aktionen, die gewünscht wird, ein. Das P gibt an, welche Ereignisse im System als Klassen möglich sind. Eine vertiefte Erklärung über WRSPM, Artefakte sowie über Phänomene können aus den Quellen ([JHLJ10], [GGJZ00]) entnommen werden.

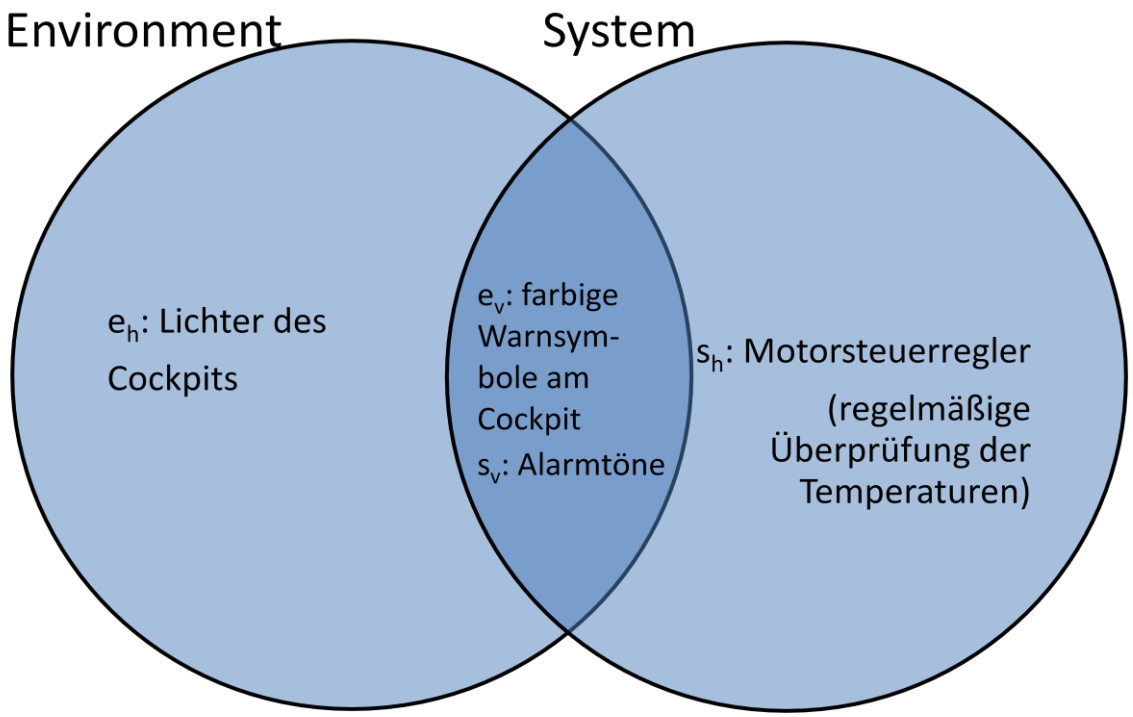

Abbildung 9: Phänomene der Artefakte (WRSPM) in der WRSPM-Methode

Die obige Abbildung macht deutlich, wie Phänomene eines Systems und einer Umgebung in der WRSPM-Methode voneinander getrennt berücksichtigt werden. Folgender Sachverhalt wird dabei dargestellt. Ein PKW-Fahrer bekommt aus dem Cockpit eines Autos Warnungen über irgendwelche Probleme oder auch Infos über die Temperatur des Motors. Falls eine Warnung unbekannt ist, so kann er das Problem verstehen, im dem er in die Bedienungsanleitung schaut. Es kann sein, dass das Öl oder das Kühlmedium nachgefüllt werden muss. Beide Medien sind notwendig für einen PKW. Falls z.B. das Kühlmedium wenig oder nicht vorhanden ist, kann der Motor nicht mehr abgekühlt werden und wird beschädigt, die je nach Typ des Fahrzeugs unterschiedlich teuer sein können. Die Umweltphänomene wären z.B. die Lichter des Cockpits mit e_h, und etwaige Warnsymbole (gelb, rot) e_v, die mithilfe der Lichter erscheinen können und gleichzeitig vom System sichtbar sind. Außerdem gibt es noch Systemphänomene, die nur z.B. vom Motorsteuergerätsystem ersichtlich sind ($s_{h)}$. Denn das Motorsteuergerät regelt z.B. auch die Kühlmitteltemperatur. In welchen Zeiteinheiten (z.B. 1s) es die Temperaturen überprüft und etwas dagegen machen muss, kann von der System-Entwicklung definiert werden. Es gibt diesbezüglich Kennfelder (im VW-System), die auch im Motorsteuergerät gespeichert sind. Sobald die Temperaturen diese Kennfelder erreichen, so wird der Thermostat durch das Steuergerät angesteuert [Wiess11]. Diese Zeiteinheit in Sekunden ist nur vom System kontrollierbar und von der Umwelt aus nicht sichtbar (kein Einfluss des Fahrers).

Das andere Beispiel, das zur Trennung der zwei Phänomen-Arten dient, ist das Herzschlagsystem der Patienten in einem Krankenhaus. Die Herzschläge des Patienten sind mit e_h, das Echo aufgrund der Herzschläge mit e_v, der Buzzer auf der Krankenschwester-Station mit s_v und die interne Darstellung der Daten durch den Sensor mit s_h gekennzeichnet. Durch diese

Beispiele ist einigermaßen klar geworden, welche Phänomene zum System und welche zur Umgebung zugeordnet werden können, wobei die Zuordnung nicht eindeutig ist. ([JHLJ10], [GGJZ00]).

Ein Open Source Softwareplattform, das als Prototyp für WRSPM entwickelt wurde und formale Strukturen aus Anforderungen darstellen kann, heißt ProR. Das Hauptziel von ProR ist die Schaffung einer Fähigkeit zur Bildung von formalen Modellen für gegebene Anforderungen. Das Tool stellt Anforderungen sequentiell dar, wobei internes Verlinken zwischen den Anforderungen möglich ist [Jast10]. Um Rückverfolgbarkeit (Traceability) zwischen dem formalen Modell und der Anforderungen, unter Einbeziehung von WRSPM, zu ermöglichen, kann ProR mit der Event-B Methode erweitert werden [JHLJ10].

3.5 Abgrenzung

In dieser Arbeit werden keine Grafersetzungssysteme wie z.B. GRGNET näher berücksichtigt, da keine Zwischenmodelle als Grafen (z.B. Omnigraf) für etwaige MX vorgesehen sind. Stattdessen wird das Eclipse EMF-Projekt zur Hilfe genommen. EMF kann strukturiertes Wissen, dass in XMI-Format bereits existiert, optimal darstellen. Das Projekt (eSALE) kann als Plugin von der Homepage des [KIT11b] für Testzwecke heruntergeladen werden. Andere Tools wie AutoAnnotator oder das eSALE-Projekt sind dort auch vorhanden. Im Rahmen des SALEMX konstruierte EMF-Diagramm, wo alle Beziehungen zwischen den Textinstanzen eines Satzes zu finden sind, wird für die Darstellung der Endergebnisse eingesetzt. Für diese Arbeit sind nur die semantische Idee von [Gelh10] und [Den07] wichtig.

Im Rahmen des Requirement-Engineering wird bei der Modellierung des Gesamtsystems das WRSPM-Umfeld mitberücksichtigt. Phänomene wurden z.B. in der Event-B Methode als Konstanten, Variablen oder Datensätze modelliert [JHLJ10]. Bei der WRPSM-Methode werden speziell, aufgrund der kompakteren Modellierbarkeit der Entitäten, immer mehrere zusammengehörige Wörter zusammen verbunden. WRSPM ist hauptsächlich für die Formalisierung der Anforderungen zuständig. Für diese Arbeit sind formale Modelle genauso wichtig, wobei linguistische Aspekte der Anforderungen im Vordergrund standen.

4 Entwicklung des „Auto SRL“ Konzepts

Wie schon im Einleitungsteil erwähnt gibt es das EMF-Projekt von Eclipse, das z.B. aus einem XMI-Dokument ein Modell erzeugt und anschließend den „ausführbaren Code“ generiert. Im Kapitel 4.2 wird dann eine MDA-Lösung via eSALE [KIT11a] und EMF vorgestellt, womit die Ergebnisse des hier entwickelten Prototyps Auto SRL weiterbearbeitet bzw. dargestellt werden sollten. Das Unterkapitel 4.3 beschreibt die Konzepte des Prototyps, das dann im Kapitel 5 implementierungstechnisch angegangen wird.

Sowohl im AA von [Land10] als auch in anderen Forschungen gibt es Ansätze, die Syntax mit der Semantik verbinden. Dieses Kapitel beschreibt ein systematisches Vorgehen, das eine Brücke zwischen der Syntax und der Semantik darstellt. Ontologien kann man als Stahlseile dieser Brücke bezeichnen, die entsprechend gepflegt und erweitert werden sollten.

4.1 Modellextraktion aus natürlicher Sprache

Im Rahmen dieser Arbeit wurde festgestellt, dass die Einbindung der natürlichen Sprache in die Software-Entwicklung ohne eine modellgetriebene Idee nicht möglich ist. Ein System (wie z.B. SALEMX), das Grundstrukturen der natürlichen Sprache semi-automatisch erkennt und daraus ein Modell (z.B. UML) extrahiert, ist hier vonnöten. Zunächst wird ein Beispiel vorgestellt, dass auf die Schwächen eines nicht systematischen Ansatzes hindeutet.

Die Abbildung 10 stellt eine Teilfunktion der Motorkühlung eines PKWs dar. Es gibt einen kleinen und einen großen Kreislauf, die für die Kühlung des Kühlmittels bzw. des Motors zuständig sind. Bei Kaltstart bzw. bei niedrigen Temperaturen des Motors ist ein kleiner Kreislauf ausreichend, wobei bei höheren Temperaturen der *große Kreislauf* geöffnet werden muss. Diese wird entweder durch den *Thermostat* (Agent) beim Erreichen einer bestimmten Temperatur oder durch *Kennfelder* (Agent), die im Steuergerät abgelegt sind, geregelt. Beim großen Kreislauf ist auch der Kühler beteiligt. Zusätzlich wird der Elektrolüfter eingeschaltet, damit bei Volllast schneller abgekühlt werden kann. Dies ist die grobe Funktionsweise eines Motorkühlungssystems, eines der in Deutschland hergestellten Autos. Der Graf in derselben Abbildung ist ein Ergebnis der semantischen Idee von [Gelh10]. Identische Rollen „AGXOR“ an zwei Knoten in derselben Relation stören zwar, können aber auch geregelt werden, sodass zur gleichen Zeit in der Relation b immer nur einer der Knoten (Thermostat, Kennfelder) die Rolle des Agenten spielt.

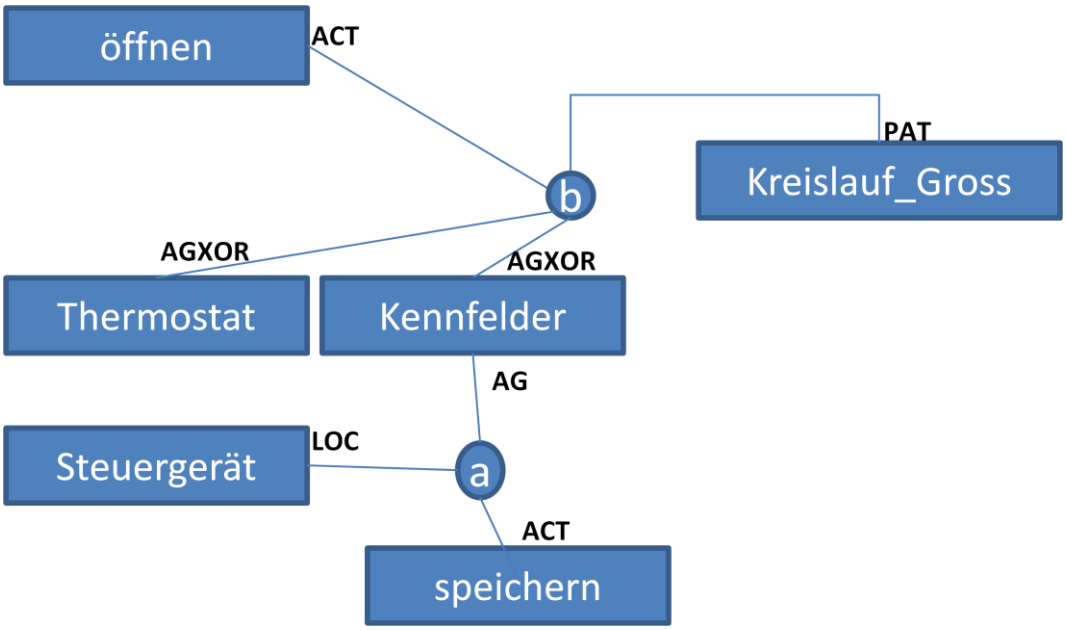

Abbildung 10: Kühlmittelumlaufsteuerung für den großen Kreislauf

Im oberen Graf ist der Sachverhalt bezüglich der Steuerung des Kühlungssystems klar dargestellt worden. Sowohl die SALE-Rollen als auch komplexe Zusammenhänge (XOR) sind nachvollziehbar. Viel wichtiger für die hier erwähnte Einbindung sind aber andere Zusammenhänge, die beschreiben, dass z.B. Wörter oder Sätze Konstituenten sind, die dann deswegen keine oder mehrere Rollen besitzen dürfen usw. Nur so kann eine rechnergestützte Modellentwicklung realisiert werden. Im weiteren Verlauf der Modellextraktion und später dann bei der Codegenerierung sollten daher der eSALE-Compiler (o.ä.) sowie das EMF (o.ä.) zum Einsatz kommen, die im nächsten Kapitel besprochen werden.

4.2 Model Driven Architecture via eSALE und EMF

Dieses Kapitel ist für die Darstellung der Endergebnisse eines rechnergestützt linguistischen Bearbeitungsprozesses zuständig. Es wird also zuerst vorgestellt, was das hier angestrebte Ziel ist, bevor der Arbeitsprozess des Auto SRL besprochen wird.

Sowohl für die Modellierung als auch für die Codegenerierung ist EMF zuständig. Das EMF steht für Eclipse Modeling Framework und unterstützt ein Teilprozess der automatischen Modellextraktion. Dies kann er nur dann machen, falls ihm strukturierte Informationen vorliegen. Um diese Struktur grafisch darstellen zu können, muss ihm auch eine Modelldatei (oder Meta-Modell) bekannt sein. Diese Datei soll bezüglich des Kontexts die gesamte Datenstruktur von SALE beinhalten. Es existiert bereits ein eSALE-Projekt [KIT11a], das genau diese Modelldatei mitliefert. Man kann Änderungen an dieser Datei vornehmen oder selbst eigene Modelldateien erstellen und mit denen ein Code generieren. Ein eSALE-Dokument ist ein XMI-Dokument, in dem Sätze, Wörter, Kommentare sowie die semantischen Rollen separat aufgelistet sind.

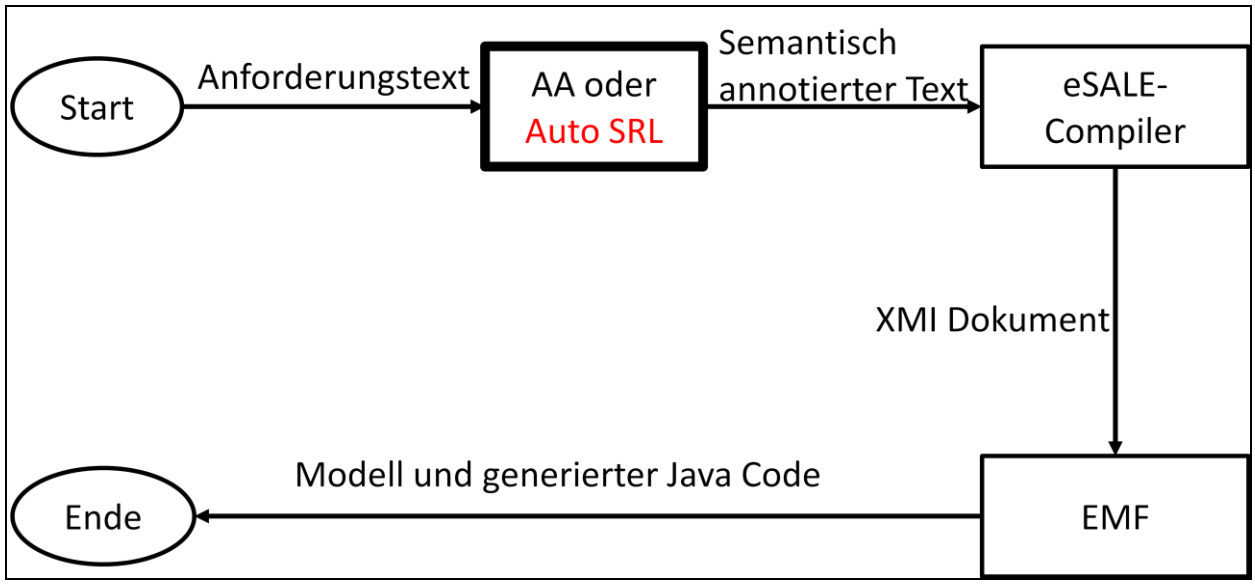

Abbildung 11: Abbildung der SALE-Syntax anhand von EMF (grobe Darstellung)

Der eSALE-Compiler erzeugt anhand von - Hand annotierten - Sätzen eine XMI-Datei. In der nächsten Abbildung ist ein Auszug aus dieser Datei zu sehen. Es handelt sich dabei um die Darstellung des folgenden Satzes durch den (e)SALECompiler: „[#The cooling_system|OMN #is #composed #of {radiator, pressure_cap AND fan}|PAR].“ Dabei geben diese Rollen an, dass das „cooling_system“ aus kleineren Teilen wie „radiator“, „pressure_cap“ oder „fan“ besteht, womit das Verb nicht mehr gebraucht wird, überflüssig ist und als Kommentar festgelegt wird. Das OMN steht nach SALE für das Ganze, wobei PAR für den Teil eines Ganzen steht. Das XMI-Dokument ist gültig, da sie z.B als root Element den Tag </eSALE:Document> enthält. Sie enthält alle Informationen über jegliche Instanzen vom Satz. Sogar Kommentare dieses Satzes (#The, #is, #composed, #of) sind in dieser Datei separat aufgelistet. Die anderen drei Wörter, die keine Kommentare sind, werden separat angezeigt. Für jede Rolle wird hierbei jeweils angegeben, zu welchem Wort sie angehören und an welcher Position sie im jeweiligen Satz vorkommen. Dieses XMI ist gleichzeitig die Eingabedatei von EMF (siehe Tabelle 13). In der übernächsten Abbildung 12 sieht man nun die endgültige Ausgabe, die durch das EMF-Tool beim Laden des XMI-Dokuments erzeugt wurde.

Tabelle 13: XMI-Datei für den Satz „The cooling_system is composed of radiator,…“

```
    <comments value="The" position="0" singleWord="true"/>
    <comments value="is" position="2" singleWord="true"/>
    <comments value="composed" position="3" singleWord="true"/>
    <comments value="of" position="4" singleWord="true"/>
…
```

```
<words name="cooling_system" value="cooling_system"/>

<words name="radiator" value="radiator" setClosure="//@phrases.0/@roleTargetSets.0"/>

<words name="pressure_cap" value="pressure_cap" s   setClosure="//@phrases.0/@roleTargetSets.0"/>

<words name="fan" value="fan" setClosure="//@phrases.0/@roleTargetSets.0"/>

<roleSets implicit="true">

  <elements type="OMN" target="//@words.0" positionInPhrase="1"       phrase="//@phrases.0"/>

  <elements type="PAR" target="//@phrases.0/@roleTargetSets.0" positionInPhrase="5" phrase="//@phrases.0"/>

</roleSets>
```

Dieses EMF-Tool verschafft auch gleichzeitig eine Brücke zwischen der Modellierungswelt und der Codegenerierung. Der Generierungs-Aspekt wurde schon vorher erwähnt. Bei der Arbeit mit dem EMF sieht man, dass bei Änderungen der Modelldatei (Modell) nicht der komplette Code neu generiert wird. Es passiert ja, dass zwischenzeitlich der Code durch den Programmier erweitert wurde. Auf diese Stellen achtet EMF mit Hilfe von Flags besonders. Somit wurde auch gezeigt, was für einen Einfluss das Tool EMF hat, um Modelländerungen in den Code vorsichtig einzuchecken.

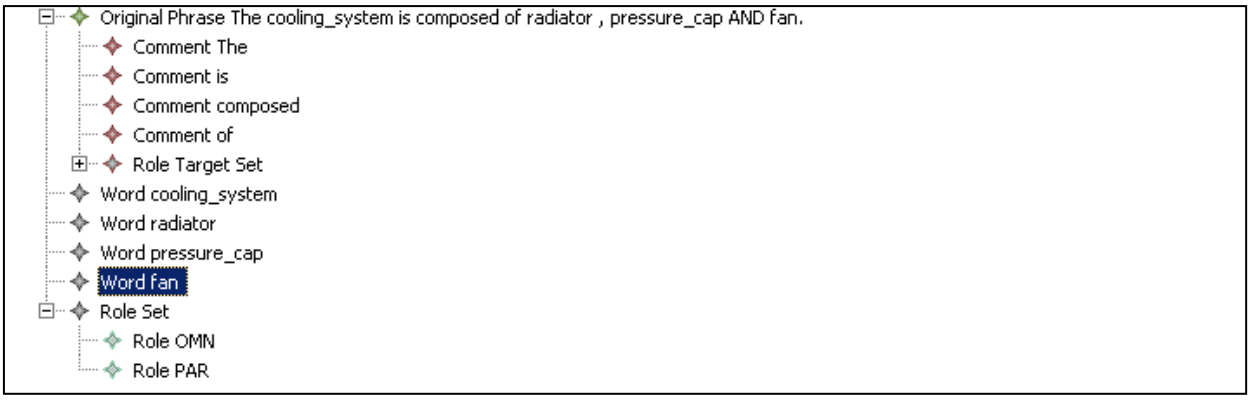

Abbildung 12: EMF-Darstellung des Satzes

Das nächste Diagramm in Abbildung 13 stellt einige Klassen des Auto SRL dar. Dieses Diagramm ist hinsichtlich der Wortartenunterscheidung nach WordNet eine kleine Ergänzung des Klassendiagramms von [Land10].

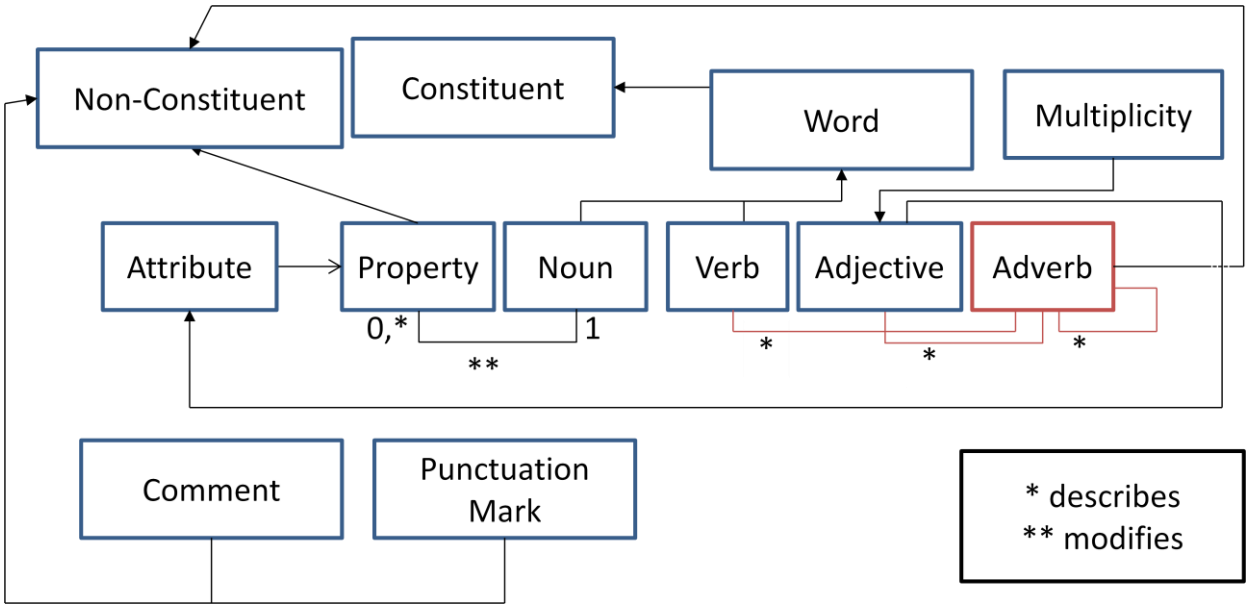

Abbildung 13: Ergänztes Klassendiagramm (Ergänzung zu [Land10])

In der Adverb.java befindet sich eine Methode ArtofAdverb(), die für die Ermittlung der Adverb-Art zuständig ist. Falls erforderlich können weitere Klassen (wie z.B. Adverb_Frequency.java) von dieser Klasse erben. Andererseits gibt es auch eine Methode „describes“, die bei Adverb.java für die Beschreibung von Verben dienen soll. Die Methode Property.java (bzw. auch in Adjective.java) modifiziert Namen. Wir profitieren also von der Idee des OOP-Paradigmas, die nicht nur die Schreibarbeit (wie unvermeidbare Wiederholungen) erleichtert sondern eine fallbezogen durchdachte Datenstruktur, die neben den Daten auch Methoden besitzt. Letztendlich beschreibt diese Datenstruktur wie die Objekte (Konstituenten, Punkte, Kommas, …) der natürlichen Sprache miteinander verknüpft sind. Im Prototyp sind nicht alle Klassen zum SALE-Modell passend, weswegen auch kein gültiges SALE-Dokument entsteht. Daher funktioniert auch der eSALE-Compiler nicht immer mit der Ausgabe dieses Prototyps.

4.3 Konzept für Auto Semantic Role Labeler (Auto SLR)

Der bereits erwähnte Auto Annotator (AA) hat die gleiche Aufgabe der Semantik-Extraktion wie der hier entwickelte Auto SRL. Die Unterschiede zum AA sind, dass hier anstatt OpenCyc die Ontologien VerbNet und FrameNet eingesetzt wurden. Als gemeinsame Komponente haben sie z.B. das WordNet und den Stanford eingesetzt. Beide Werkzeuge erhalten als Input einen Plain-Text (.txt), der als Output einen SALE annotierten Text (.SALE) ausgibt, wobei der Auto SLR sich nicht unbedingt an die SALE-Grammatik hält, sondern andere Symbole ($$ z.B. für Adverbien) oder auch andere Rollen wie z.B. vom VerbNet oder vom FrameNet zu lässt. Öfters wurden diese mit SALE-Rollen ersetzt. Zum Anfang wird in AUTO SRL wie in AA alle Wörter mit dem SALE-Symbol # kommentiert. Danach beginnt der Prozess der Erkennung bzw. Annotation, in dem das #-Symbol schrittweise entfernt wird. Dieser Prozess bringt natürlich auch einige Probleme mit sich mit. Bei der automatischen Erstellung solcher Annotationen ist es

z.B. schwierig mehrere zusammenhängende Sätze zueinander zuzuordnen, falls keine Verbindungswörter (z.B. um, dann, …) erkannt werden können. Der AA versucht diese und andere Probleme der Texterkennung zu lösen. Hier werden aber nur einige Wörter erkannt, die zwei Teilsätze verbinden könnten. Wie viele Arbeiten zu diesem Thema schon zeigen, ist es schwierig eine allgemeingültige Lösung für die genannten Probleme zu finden. Die Aufgabe dieser Arbeit ist es, potentielle Lösungswege zu erkennen und anschließend diese sinnvoll durchzuführen. Diesbezüglich wurden speziell technische Sätze - die nicht unbedingt einfach sind - getestet und anhand diese mögliche Schritte zu Lösung des Problems erkannt. Die Beispielsätze finden sich im Anhang D.

Diese Tests wurden sowohl mit dem Stanford-Parser als auch mit dem AA parallel durchgeführt. Der Stanford-Parser kann über die Homepage benutzt werden. Gleichzeitig kann man sie auch als API in Java-Anwendungen integrieren und somit losstarten. Die Arbeitsweise des AA, das ein Eclipse-Plug-In ist, wird ausführlich in [KIT11a] bzw. in [Land10] erklärt. Zum Starten des AA, gibt es zwei Möglichkeiten. Man kann es entweder über den Eclipse oder über die Konsole mit Parametern starten. Weitere Details dazu siehe Anhang A. In den weiteren Unterkapiteln dieses Kapitels wird die Arbeitsweise des AUTO SRL vorgestellt.

4.3.1 Erkennung der Teilsätze und ggf. Bestimmung ihrer Rollen

Neben einigen Signalwörtern (Keys), die für die Ermittlung der Grenzen oder Rollen von Sätzen bzw. Teilsätzen eine wichtige Rolle spielen, gibt es im Stanford Hinweise, die die gleichen Informationen liefern könnten. Zur Veranschaulichung dieses Sachverhalts werden nun einige Ergebnisse des Stanford Parsers vorangestellt. Zwischen den Teilsatzgrenzen soll mindestens ein Verb erscheinen. Der untere Baum (Tabelle 15) stellt den Satz strukturiert dar, wobei Teilsätze durch S oder SBAR erkannt werden können. Der POS-Tag „IN“ mit „at“ in der 14. Zeile deutet auf eine Präposition hin. Ein anderer Fall, wo „IN“ eingesetzt wird, sind konjunktionale Nebensätze. Also kann der IN-Tag sowohl auf eine Präposition als auch auf einen konjunktionalen Nebensatz hindeuten. Der zweite Fall ist zusammen mit dem „S“ der interessantere Fall, da dies auf einen Satzanfang eines Teilsatzes hinweist.

In der Zeile 14 der Tabelle 16 gibt es einen Hinweis, dass der Satz eine koordinierte Konjunktion „CC„ enthält. Zu den meist verwendeten CCs gehören vor allem: and, or, either…or etc. Durch einen CC-Tag entstehen zwei separate Sätze. Bei einem Preorder-Durchlauf des durch den Stanford Parsers ausgegebenen Baumes können die „Satzglieder angrenzende“ Knoten ermittelt werden. So kann man im unteren Baum die Teilsätze zwischen dem ersten und dem zweiten „S“, zwischen zweitem „S“ und dem Komma und schließlich den Teilsatz zwischen dem „CC“ und dem Punkt erkennen. Man sieht, dass der Satz aus drei für Satzerkennung interessanten Bereichen besteht, da alle Bereiche mindestens ein Verb beinhalten. Nun kann die Frage auftauchen, warum man den Satz in Teilbereiche aufteilt und dann weiter analysiert (z.B. Ermittlung der lokalen Dependencies). Der Stanford Parser kann den Satz komplett bearbeiten aber er ist leider nicht für die semantische Rollenvergabe zuständig. Der stellt wie andere Parser nur die syntaktischen Abhängigkeiten sowie die Wortarten eines Satzes fest und gibt diese aus.

Tabelle 14: Baumdarstellung Satz 1 (Stanford Parser)

```
0      (S
1        (NP (NN Gasoline) (NNS engines))
2        (VP (VBP have)
3          (VP (VBN improved)
4            (NP
5              (NP (DT a) (NN lot))
6              (, ,)
7              (SBAR
8                (S
9                  (NP (PRP they))
10                 (VP (VBP are)
11                   (ADVP (RB still))
12                   (RB not)
13                   (ADJP (RB very) (JJ efficient)
14                     (PP (IN at)
15                       (S
16                         (VP (VBG turning)
17                           (NP (JJ chemical) (NN energy))
18                           (PP (IN into)
19                             (NP (JJ mechanical) (NN power)))))))))))))
```

Tabelle 15: Baumdarstellung Satz 2 (Stanford Parser)

```
0     (ROOT
1      (S
2        (NP (DT The) (JJ cooling) (NN system))
3        (VP
4          (VP (VBZ allows)
5            (NP (DT the) (NN engine)
6              (S
7                (VP (TO to)
8                  (VP (VB heat)
9                    (PRT (RP up))
10                   (ADVP (RB as) (RB quickly))
11                   (PP (IN as)
12                     (ADJP (JJ possible)))))))))
13         (, ,)
14         (CC and)
15         (VP
16           (ADVP (RB then))
17           (VBZ keeps)
18           (NP (DT the) (NN engine))
19           (PP (IN at)
20             (NP (DT a) (JJ constant) (NN temperature)))))
21       (. .)))
```

Im nachfolgenden werden Schlüsselwörter vorgestellt, die für die Ermittlung der Haupt- und Nebensatz-Rollen wichtig sind. Diese Key-Wörter sind aber für den Auto SRL nicht ausreichend um die Kontrolle über alle Nebensätze zu besitzen. Im AA gibt es auch wenige solcher Schlüssel, die mit dem Präfix „signal“ hinterlegt sind. Die Key-Wörter mit ihren jeweiligen Rollen sind dort folgendermaßen hinterlegt: because mit CAU, to mit INT, if mit

SUM und while mit TEMP. Die zugehörige Rolle in Tabelle 17 (Spalte 3) kodiert schon die Semantik der Key-Wörter, so dass diese eigentlich kommentiert bleiben könnten. Da aber z.B. „when“ im zweiten Satz sich auf das Verb „melts“ adverbial bezieht, darf „when“ nicht als Kommentar bleiben. Bei when handelt es sich also um ein Adverb, genauer um ein Wh-adverb, sodass es durch den Stanford-Tagger zu den WRB-Tags eingeordnet wird und die Rolle des Teilsatzes der SUM entspricht. Die andere Rolle TORIG in der Tabelle 19 deutet auf einen Zeitpunkt hin, seit dem etwas sein wird. Auf das Beispiel bezogen sollte nach einem bestimmten Temperaturstand (Key-Wort:„Once“) etwas passieren.

Tabelle 16: Satz- oder Teilsatzerkennung durch Schlüsselwörter 1

Satzteil-Erkennung	**Beispielsätze aus (Anhang D) & passende Rollen nach [Gelh10]**
Konjunktionaler Nebensatz. when = WRB-Tag, then = RB-Tag + ADVP	(Satz 1) … and they turn on when the temperature goes above a set point. (CC and) (S (NP (PRP they)) (VP (VBP turn) (PP (IN on) (SBAR (WHADVP (WRB when)) (S (NP (NP (DT the) (NN temperature)) (PP (IN of) (NP (DT the) (NN coolant)))) (VP (VBZ goes) (PP (IN above) (NP (DT a) (VBN set) (NN point))))))))) (. .))) (Satz 2) When the wax melts, … (S (SBAR (WHADVP (WRB When)) (S (NP (DT the) (NN wax)) (VP (VBZ melts)))) (, ,) Role → **SUM**

Tabelle 17: Satz- oder Teilsatzerkennung durch Schlüsselwörter 2

Satzteil-Erkennung	**Beispielsätze aus (Anhang D) & passende Rollen nach [Gelh10]**
Konjunktionaler Nebensatz so that = RB & IN & SBAR	(Satz 4) The cooling fan has to be controlled so that it allows … (S (NP (DT The) (VBG cooling) (NN fan)) (VP (VBZ has) (S (VP (TO to) (VP (VB be) (VP (VBN controlled) (SBAR (RB so) (IN that) (S (NP (PRP it)) (VP (VBZ allows))))))))))) Role → **CONS+**

Tabelle 18: Satz- oder Teilsatzerkennung durch Schlüsselwörter 3

Satzteil-Erkennung	**Beispielsätze aus (Anhang D) & passende Rollen nach [Gelh10]**
Konjunktionaler Nebensatz once = IN & SBAR as soon as = ADVP then = RB	(Satz 9) Once the temperature of the coolant rises to between … (SBAR (IN Once) (S (NP (NP (DT the) (NN temperature)) (PP (IN of) (NP (DT the) (NN coolant)))) (VP (VBZ rises) (PP (TO to) (NP (QP (IN between) (CD 82) (CC and) (CD 91)) (NNS Celcius)))))) Role → **TORIG**

Tabelle 19: Satz- oder Teilsatzerkennung durch Schlüsselwörter 4

Satzteil-Erkennung	**Beispielsätze aus (Anhang D) & passende Rollen nach [Gelh10]**
Nebensatz mit Relativ-Pronomen which = WDT that = WDT whichever = WDT who = WP	(Satz 7) The cylinder which is located on the engine side ... (S (NP (NP (DT The) (NN cylinder)) (, ,) (SBAR (WHNP (WDT which)) (S (VP (VBZ is) (VP (VBN located) (PP (IN on) (NP (NP (DT the) (NN engine-side)) (PP (IN of) (NP (DT the) (NN device))))))))) (, ,)) Role → - (Satz 7) ... is filled with a wax that begins ... (VP (VBZ is) (VP (VBN filled) (PP (IN with) (NP (NP (DT a) (NN wax)) (SBAR (WHNP (WDT that)) (S … Role → -

Zusammengefasst können folgende Ableitungsregeln für die Bäume der Stanford Parser ermittelt werden:

1. Der Knoten S spannt immer einen Baum auf. Dieser Baum kann den vollständigen Satz oder auch einen Teilbereich davon darstellen. Im ersten Fall enthält der S –als Kind des Roots- einen (. .) als End-Blattknoten (globaler S) und im zweiten Fall keins (lokaler S).

2. Alle S-Knoten können innerhalb ihres aufgespannten Baumes eine oder mehrere CC-Konjunktionen, Kommas, Semikolons, oder auch SBARs als Kind-Knoten beinhalten, die wiederum einen Teilbaum aufspannen können.

3. Ein SBAR-Knoten kann gleichzeitig S als Kind-Knoten beinhalten. Beispielsweise kann hier folgende Regel auftreten: SBAR → Adverb + S.

4. CC gilt für S nur dann als Konjunktion, falls sie auch gleichzeitig einer der ersten Kinder von S ist.
5. Eine Verbalphrase kann auch Knoten analog zu 2. enthalten.
6. Eine Nominalphrase besitzt auch dieselben Knoten wie in 5. außer von SBAR und S.

Konjunktionale Nebensätze in der Tabelle 21 und Relativ Pronomen in der Tabelle 22, die in den nächsten zwei Tabellen aufgelistet sind, helfen bei der Ermittlung von Nebensatzgrenzen, sodass wenn diese auftauchen automatisch Teilsätze gefiltert werden können.

Tabelle 20: Nebensatz: Schlüssel-Wörter für Konjunktionale Nebensätze (siehe Anhang B)

after	once	Until
although	provided that	When
as	rather than	whenever
because	since	…

Tabelle 21: Nebensatzerkennung: Schlüssel-Wörter für Relativ Pronomen

that	who	whose
which	whoever	whosever
whichever	whom	whomever

Da es aber davon – Relativ Pronomen oder Schlüsselwörter für konjunktionale Nebensätze - noch mehr gibt, die hier nicht alle berücksichtigt werden können, werden die hier aufgelisteten zu den Defaults zugeordnet. Ansonsten kann man sich nach den obigen Ableitungsregeln (z.B. für S oder SBAR) des Stanford-Baumes richten um einen Teilsatz zu bestimmen. Zu den speziellen Wortarten für konjunktionale Nebensätze könnte man die Adverbien zählen, die im nächsten Kapitel behandelt werden.

4.3.2 Relevante Ontologien für Auto Semantic Role Labeler

Die bisherigen Erfahrungen machen deutlich, dass allein der Parser nicht ausreicht sowohl syntaktische als auch semantische Informationen über den Text zu liefern. Neue Komponenten

müssen eingesetzt werden, damit semantisch verfahren werden kann. Zu den großen Ontologien gehören der Cyc sowie die Unified Verb Index. AA hat den Cyc benutzt um einige unbekannte Rollen für Konstituenten festzustellen. Das Cyc bietet mehr Konzepte oder Assertions (Fakten und Regeln) als eine andere Ontologie wie z.B. SUMO, die im Kapitel lexikalische Ressourcen behandelt wurde, an. SUMO kann man auch speziell für domänenspezifische Name-Erkennung (z.B. in „communication“, „distributed computing“, … siehe Kapitel 2.2) einsetzen. Die Entscheidung, welche Ontologie oder Ontologien letztendlich eingesetzt wird, ändert sich vom Fall zu Fall. Die nächste Abbildung gibt eine Übersicht warum in dieser Arbeit manche Ontologien eingesetzt und andere nicht eingesetzt wurden.

Eigenschaften/Ontologien	Cyc	SUMO	VerbNet	FrameNet	WordNet
Java API vorhanden?	Ja	Nein	Ja	Ja	Ja
Ontologie enthält Stanford ähnliche Muster wie zum Beispiel: NP VP PP	-	-	Ja	Ja	-
Ontologie wurde im AUTO SRL eingesetzt für	-	-	Erkennung von Verbstrukturen	Semantik-Extraktion anhand Verb, Adverb und Adjektiv	Ermittlung der Grundform von Wörtern

Abbildung 14: Übersicht über relevante Ontologien (außer Stanford Parser)

4.3.3 Bestimmung von Rollen durch Adverbien

Da es von den Adverbien wenige Arten gibt, kann man für Teilsätze oder Wörter die Rollenzuweisung besser wie für die obigen Schlüsselwörter realisieren. Wie aus dem Anhang B entnehmbar, gibt es einige Möglichkeiten, wo sich die Adverbien (RB-Tags des Penn-Tagset) positionieren können. Adverbien können die Adjektive, Verben oder auch andere Adverbien beschreiben, wobei Adjektive nur Namen (NN-, NNS-, NP-, NPS-Tag) modifizieren [PMZ11b]. Auch in vielen Literaturen steht, dass Namen und Attribute (als Adjektiv) in Kombination stehen.

Stanford unterstützt die Adverb-Erkennung sowohl mit seinen Dependencies als auch mit seinen Tags. Der advmod(b,a) gibt an, dass „a“ ein „b“ beschreibender Adverb ist. Beim „b“ handelt es sich um ein Verb. Ein anderer Fall, dass „b“ ein Adjektiv oder was anderes ist, kam bei den getesteten Beispielen nicht vor und wird nicht behandelt. Jetzt sollte man anhand des advmod ermitteln, an welcher Position das Adverb „b“ steht. Steht sie am Anfang des Satzes (z.B. when), handelt es sich um ein Zeitadverb, der gleichzeitig zwei Sätze miteinander verbinden

kann(z.B. then). Kommt es nach dem Verb oder nach dem direkten Objekt, so kann es eventuell ein Ortadverb sein. Diese sind aber allgemeine Regeln der englischen Grammatik, die bei der automatischen Verarbeitung nicht trivial sind. Um die Sache noch eindeutiger zu machen hilft FrameNet, das angibt um welche Art von Adverb (z.B. das Frame „frequency“) es sich hier handelt. Man erkennt, dass die Adverb-Art „Häufigkeit“ eindeutig identifiziert werden kann, wobei die anderen gerade noch abgeschätzt werden können, da mehrere Frames von FrameNet angeboten werden. Nicht präzise sieht die Sache bei der Adverb-Art der Art und Weise aus. Hierfür gibt es im FrameNet leider sehr viele Frames, die die Sache erschweren. Außer der letzten Art von Adverbien kann ein Umweg über das WordNet helfen, um Wörter zu bestimmten Frame-Kategorien zuzuordnen. Falls das betrachtete Adverb in einem für Auto SRL unbekanntes Frame liegt, so soll geschaut werden, ob vielleicht ein „starkes“ Synonym zu diesem Wort nicht in einem bekannten Frame liegt. Ist dies der Fall wird das Synonym-Frame genommen, ansonsten wird das gleiche Verfahren mit einem anderen Synonym aus WordNet wiederholt. Einige Frames (Tabelle 21) sollten aber dem Auto SRL bezüglich der Adverbien bekannt sein, damit dieser Trick funktioniert. Dies ist wiederum eine Vernetzung zweier Ressourcen (hier Ontologien), wie auch davor mit Stanford und dem FrameNet passiert ist. Solche Kombinationen sind in mächtigen Problemen notwendig, da eine Ressource nie alle Informationen beinhalten kann. In der nächsten Tabelle sieht man z.B., dass drei Frames zu den Zeitadverbien zugeordnet werden können.

Tabelle 22: Bestimmung der Art der Adverbien über FrameNet

Wort	**Adverbart nach FrameNet**	**Adverbart**
always, never, seldom, usually	frequency	Häufigkeit
recently, now, then , when	Temporal collocation	Zeit
yesterday	calendric unit	Zeit
Afterward	Time vector	Zeit
here, there	locativ relation	Ort
slowly, carefully	Taking time, Mental Property	Art und Weise

Es kann sein, dass die Adverbien für eine Semantik-Extraktion zuerst überflüssig erscheinen. Aber mit Hilfe dieser Spezialisierung in Zeit, Ort, Häufigkeit oder der Art und Weise kann man die gewonnenen Informationen später bei der Frame-Suche oder sogar bei der Entwicklung von

Frames (z.B. für FrameNet) als Zusatzinformation einsetzen. In dieser Arbeit wurden sie aber speziell für die Verfeinerung der Semantik eingesetzt.

Da nach SALE alle Konstituenten eine Rolle haben sollten, bekommen nur Wörter und Sätze eine Rolle. Kriterium für eine Konstituente ist, dass sie ein Spektrum aufweisen. Nach [Gelh10] sollten also nur Adjektive (da sie zu geschlossene Wortklassen gehören; siehe auch Tabelle 13) aber Adverbien keine Rollen bekommen. Da aber sowohl Adjektive als auch Adverbien wiederum nach der SALE-Grammatik Subordinenten sind, bekommen sie keine Rollen und werden mit „Shifts“ und Ziffern (siehe Kapitel 4.1.1) markiert. Adverbien können neben der Beschreibung eines Verbes die Rolle des fein-granularen Nebensatzes bzw. Satzes bestimmen und werden hier daher zusätzlich mit $$ anstatt $ gekennzeichnet. Somit können auch Adverbien von den Adjektiven unterschieden werden.

4.3.4 Bestimmung von Rollen durch Verben oder Adjektive

In den Grundlagen wurde behandelt, dass in VerbNet anhand von Verben und in FrameNet anhand von Prädikaten, substantivierten Verben sowie von Adjektiven die Ontologien erstellt wurden. FrameNet macht bei jedem Lexical Unit durch einen Postfix deutlich, dass „x.v“ ein Verb, „x.a“ ein Adjektiv und „x.n“ ein substantiviertes Verb ist. Diese LU’s gehören einem oder mehreren Frames an, die jeweils Reports über lexikalische Einträge sowie Annotationen enthalten. In der Abbildung 15 wird gezeigt, wie FrameNet und VerbNet Information an den Reasoner des Auto SRL liefern. Als nächstes werden Beispiele vorgestellt, die mit den Frames vom VerbNet gleiche Strukturen aufweisen.

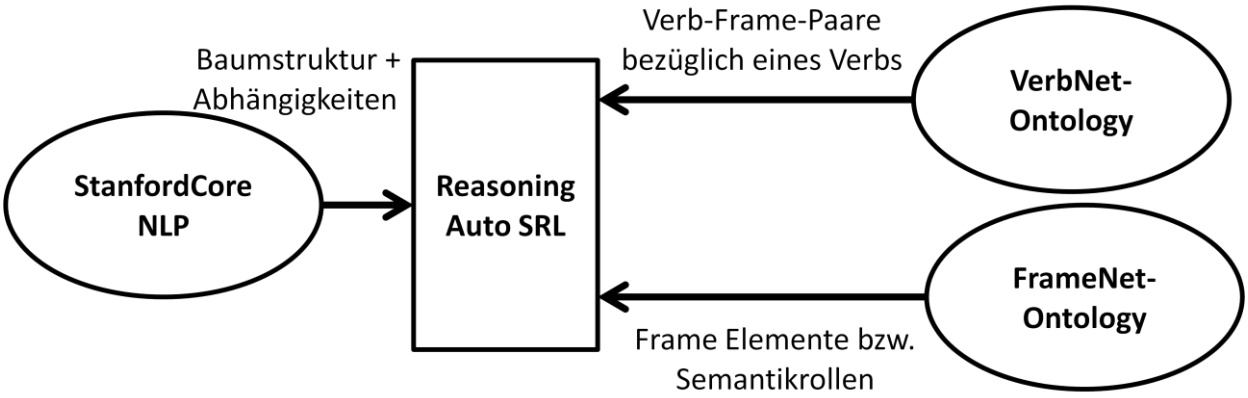

Abbildung 15: Informationslieferanten für Auto SRL (außer WordNet)

Ein vollständiger Satz beinhaltet mindestens ein Verb worauf man sich beim automatischen Prozess verlassen kann. Um Informationen über das Verb herauszuholen wurde im Auto SRL das VerbNet als eine relevante Ontologie angesehen, da sie Strukturen der Form NP V [NP|PP] oder der ähnlichen besitzt, die den Strukturen des Stanford Parsers ähnelt. Im vorherigen Kapitel hat man Teilsätze weitestgehend fein-granular bestimmt – eine mühselige Arbeit – um dann die Rollen weitestgehend präzise vergeben zu können. Dabei gewonnene Informationen werden hier zusammen mit den Abhängigkeiten des Parser kombiniert. Dabei werden mithilfe von

Abhängigkeiten nur vom Verb relevanten Bereiche des Baumes untersucht. Die Abbildungen 16-19 verdeutlichen diese Fälle. Falls VerbNet nun die gesuchte Ontologie-Information für die Argumente des betrachteten Verbs enthält, ist durch das VerbNet eine direkte Rollenübernahme zur Baumstruktur des Stanford-Parsers ohne weiteres möglich. Ein kleines Problem stellt dabei die Suche eines geeigneten Frames im VerbNet, die in den weiteren Kapiteln optimiert werden soll.

Wie bekannt werden als erstes die Satz- bzw. Teilsatzgrenzen bestimmt. Neben diesen Informationen werden gleichzeitig Dependencies wie „nsubj()“ oder „prep()“ benötigt, die die gewünschten Stellen im Baum am schnellsten angeben können. Das Problem wäre gelöst, falls es sich hier nur um einen Verb handeln würde. Falls mehr als zwei Verben in einem Teilsatz bzw. Satz oder in mehreren Teilsätzen, die voneinander abhängig sind, existieren, so müssen andere Abhängigkeiten herangezogen werden. Einer davon ist z.B. die Abhängigkeit „conj(a,b)“. Das erste Verb (hier a) steht dabei mit dem anderen Verb (hier b), das sich in einem anderen Teilsatz des gleichen Satzes befindet, in Beziehung. In diesem Fall verweist b auf das gleiche Subjekt wie der von a. Beide verweisen auf das gleiche Subjekt. Damit ein Vergleich mit VerbNet stattfinden kann, muss immer ein NP vorangestellt sein. Dieses ist ein Hinweis dafür, dass jedes betrachtete Verb ein ihm zugehöriges Subjekt (direkt oder indirekt) besitzt. Je mehr Verben mit dem Verb a was zu tun haben, desto mehr Abhängigkeiten mit „conj(a,x)“ werden durch den Parser ausgegeben. Die „conj()“-Abhängigkeiten helfen also indirekt den passenden Subjekt (bzw. die Nominalphrase) für das Verb (bzw. für die Verbalphrase) zu finden.

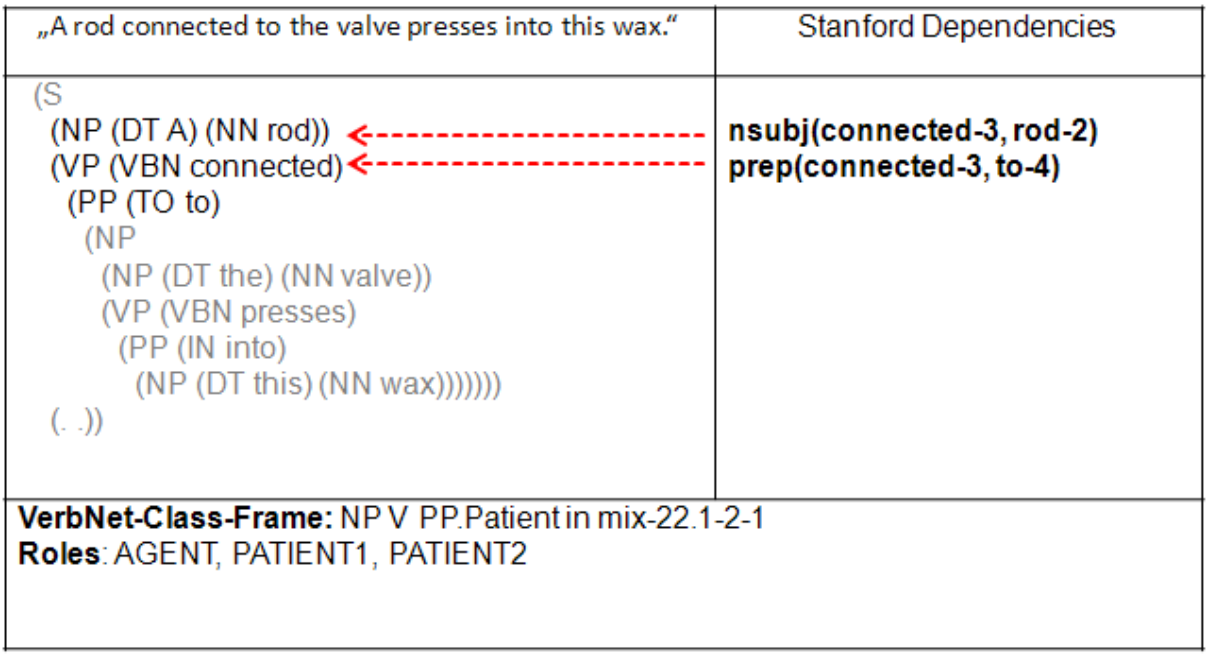

Abbildung 16: Erkennung der Verb-Abhängigkeiten für „connect“

Wie man in der Abbildung 16 sieht, wird nur der Teilbaum (bzw. Teilsatz) berücksichtigt. Diese Vorbereitungen werden durchgeführt, damit später ein Vergleich der Daten mit Ontologie-Daten

möglich ist. Als nächstes wird der andere Teil des gleichen Satzes in Abbildung 17 betrachtet, wobei hier der Parser etwas übersieht und weswegen er einen Fehler macht. Das Wort „presses“ wird als ein Name in Plural (NNS) verstanden, womit ein unvollständiger Satz nach der Präposition „to“ entsteht. Falls hier der WordNet die einzelnen Wörter bearbeitet, so gibt er an, dass das Wort „presses“ sowohl als Name als auch als Verb vorkommen kann. In so einer Konfliktsituation kann eine Benutzereingabe hilfreich sein. Hier wird eine selbst schlussfolgernde Methode entwickelt, die im nächsten Kapitel behandelt wird und dabei Ontologie-Informationen benutzt. Hier sieht man mit welchen schwierigen Fälle Parser konfrontiert sind. Nach einer erfolgreichen Ermittlung als Verb kann nun über VerbNet das passende Muster für „presses into“ ermittelt werden. Hierbei weiß man über die Dependency „nsubj (connected, rod)“, dass „rod“ der Agent dieses Satzes ist und in dem Fall (nsubj) den Agenten spielt. Die Sache sieht aber im Fall (nsubjpass) anders aus, wobei hier der Agent zum Objekt wird.

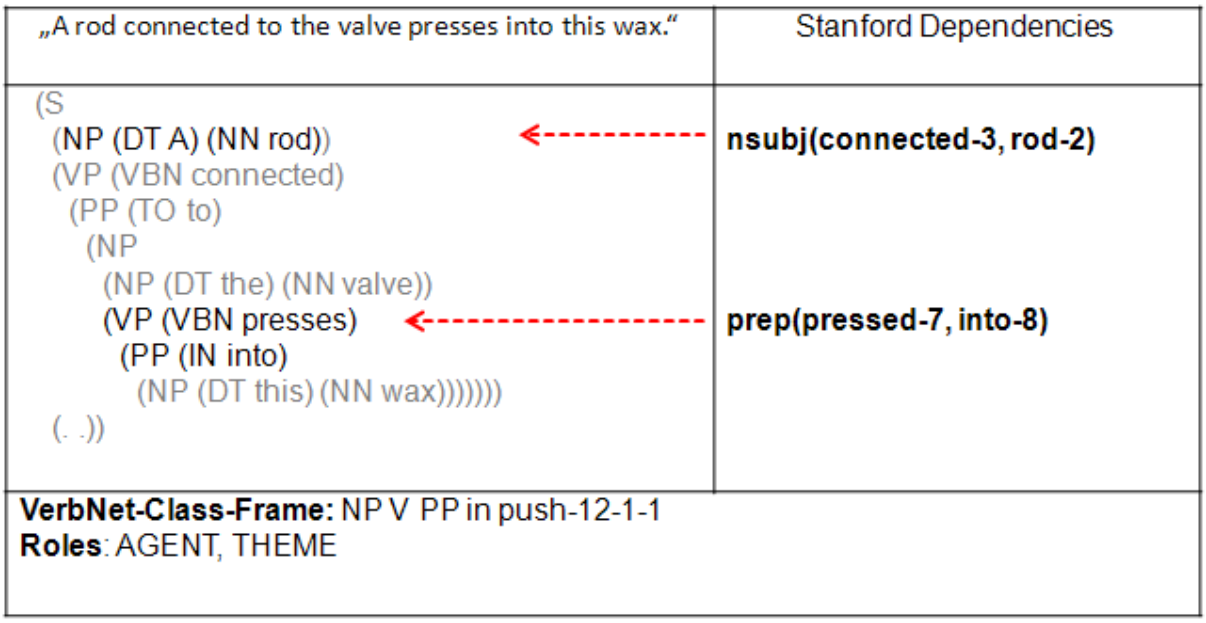

Abbildung 17: Erkennung der Verb-Abhängigkeiten für „presses“

Im nächsten Beispiel wurden auch zusätzlich syntaktische sowie semantische Informationen aus VerbNet mit angegeben. Diese wurden in dieser Arbeit nicht berücksichtigt. Zwar sind „listen“ und „search“ zwei unterschiedliche Wörter, können von der Struktur her in das untere Frame in rummage-35 eingeordnet werden. Beide beinhalten die gleiche FrameStruktur, weil sie die Präpositionen PP.Location und PP.Theme besitzen. Semantisch interpretiert erfolgt in dem Zeitpunkt, wo dieses Ereignis passiert folgendes: Der Agent - Antreiber der ganzen Sache- sucht an einem Ort (hier Tür) nach etwas (hier Ball). Im Whois_Server Beispiel hört bzw. wartet der Server am Port 43 auf die Anfragen der Clients. Wie man sieht passen beide Beispiele in das gleiche Frame mit der Struktur „NP V PP.Location PP.Theme“ rein.

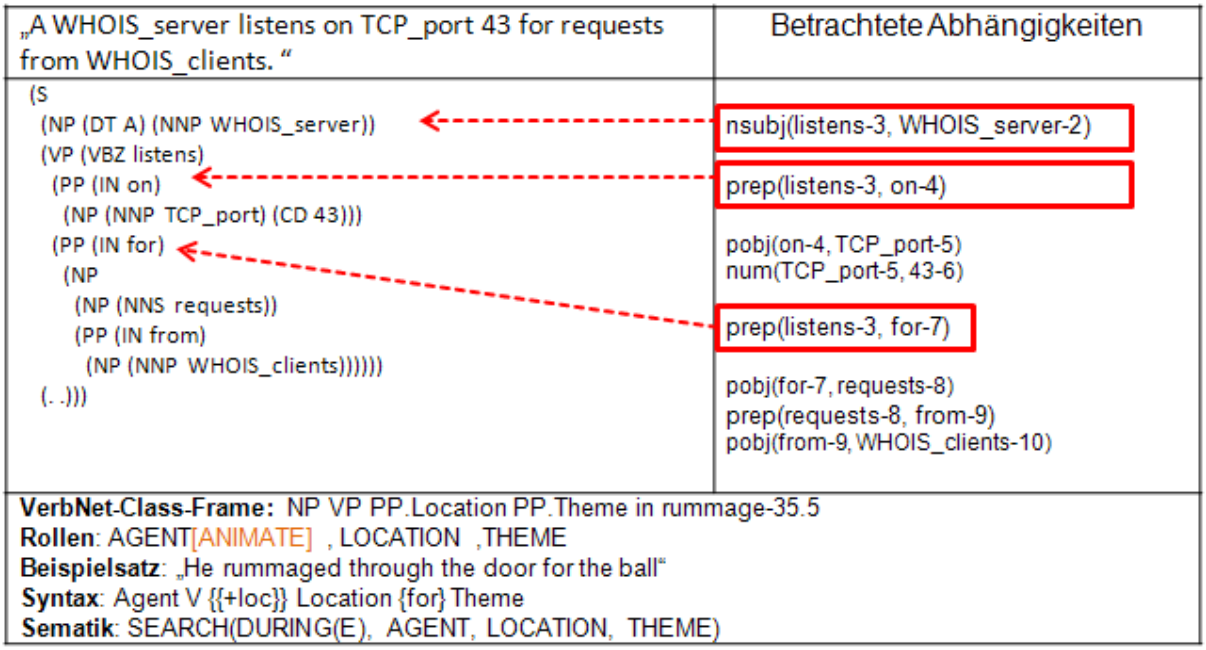

Abbildung 18: Erkennung der Verb-Abhängigkeiten für „listens“

Nun folgt ein Beispielsatz, dessen Semantik über Strukturen des FrameNet erkannt werden soll. Wie anfangs erläutert gibt es für jede Lexikalische Einheit zwei Arten von Report-Möglichkeiten. Im Lexical Entry (LE) von located.a –Verb als Adjektiv- gibt es Patterns, die Hinweise über mögliche Kombinationen dieses Verbs mit anderen Satzgliedern geben. Eines der Muster enthält „Location“ und „Theme“ als syntaktische Frameelemente, wobei das erste mit einer Präposition PP[on] hinterlegt ist. Das andere Frameelement „Theme“ entspricht der Rolle der NP. Ähnliche Hinweise gibt es auch im Annotation-Report der FrameNet-Homepage. Erfreulicherweise werden im LE -Report Frame-Elemente gewonnen, die zwar in FrameNet als syntaktische Muster gekennzeichnet sind, aber als neue semantische Rollen im Auto SRL übernommen werden können. Bei den Frame Elementen wird auch zwischen Kern und nicht KernElementen unterschieden. Im unteren Beispielsatz kommen nur die Kern FE vor. Die primäre Aufgabe bezüglich FrameNet ist nicht direkt nach FEs zu suchen, sondern FE indirekt anhand über syntaktische Strukturen z.B. durch Abhängigkeiten aus dem Stanford Parser zu erkennen.

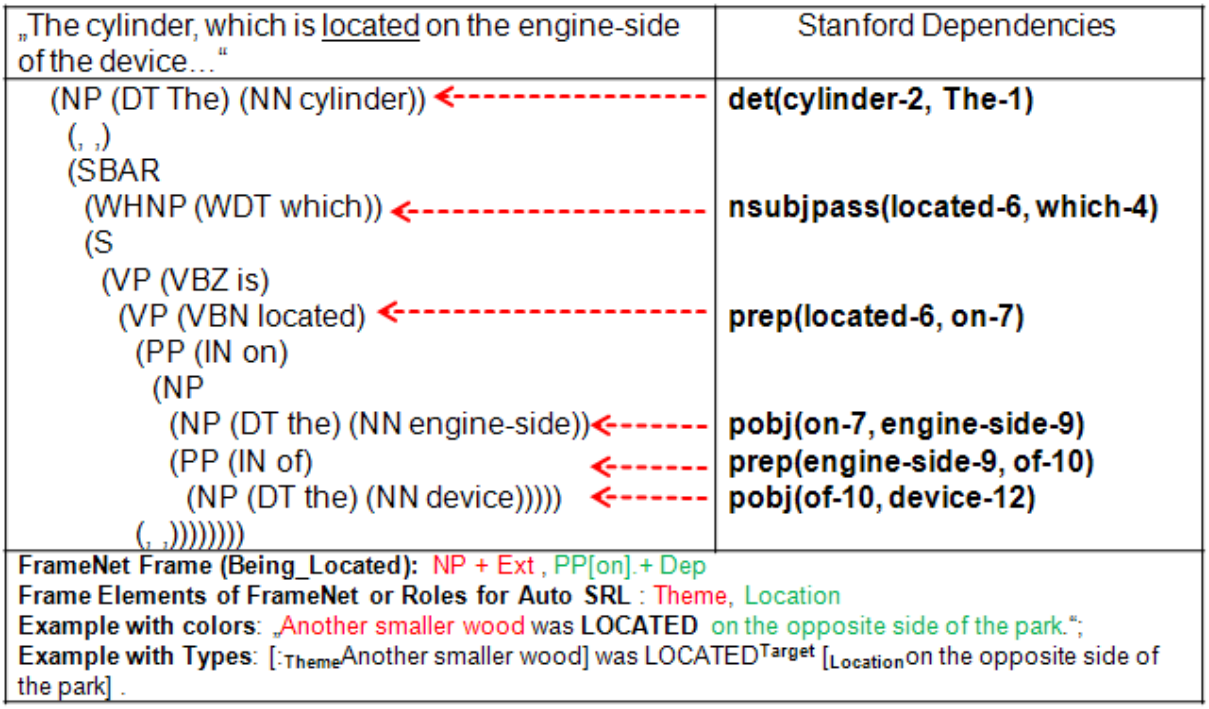

Abbildung 19: Erkennung der Verb-Abhängigkeiten für „Located on“

Der obige Beispielsatz aus Abbildung 19 wird in der Abbildung 20 mit einem anderen Satz, der mit einer gegebenen Struktur in FrameNet vorkommt, mit Hilfe von Stanford Parser verglichen. Beide Baumstrukturen ähneln sich sehr. Vor allem sind, wie in der unteren Abbildung zu sehen ist, die Verbalphrase von besonderer Bedeutung. Die Strukturkette (VP VP PP..NP) beider Sätze stimmen zu 100% überein. So schön es auch klingt, gelingt es nicht immer, solche passende Strukturen in dieser Genauigkeit zu finden. Daher sollen wie üblich die Abhängigkeiten mit einbezogen werden.

Die Struktur ab VP des Beispielsatzes aus dem Anhang D	Struktur des Beispielsatzes ab VP aus dem FrameNet
„The cylinder, which is located on the engine-side of the device…“	**“Another smaller wood was LOCATED on the opposite side of the park .”**
(**VP** (VBZ is) (**VP** (VBN located) (**PP** (IN on) (**NP** (**NP** (DT the) (NN engine-side)) (**PP** (IN of) (**NP** (DT the) (NN device)))))	(**VP** (VBD was) (**VP** (VBN LOCATED) (**PP** (IN on) (**NP** (**NP** (DT the) (JJ opposite) (NN side)) (**PP** (IN of) (**NP** (DT the) (NN park)))))

Abbildung 20: Vergleich zweier Baum-Strukturen

Der gesamte Ablauf ist nun fasst fertig. Es werden nur noch die Abhängigkeiten vorgestellt, die einen solchen Vergleich zwischen FrameNet und Stanford-Strukturen zulassen. Wie in dem Beispielsatz „A rod connected to the valve presses into this wax“ gezeigt wurde, sind die Abhängigkeiten „nsubj(V,NN)“ sowie „prep(V,PP)“ für einen Aktivsatz zuständig gewesen. Wie sieht es nun aus, wenn der betrachtete Beispielsatz als Passiv etwa so aussieht „Wax was pressed into by a rod.“. Es wird genauso nach einem Passivsatz aus FrameNet gesucht um die korrekte Semantik zu finden. Verb + PP-Kombinationen sind hierbei von Bedeutung (z.B. press into). Im FrameNet hat man auch für passive Strukturen entsprechende Frames wie z.B. Caus_motion Frame → press.v → 829-pass-avpback. Dieser beinhaltet den Satz „She was PRESSED back into her seat as the craft began to speed up“. Nun wird mit Hilfe des Wissens bezüglich der Teilsatzgrenzen das man aus den vorherigen Kapiteln besitzt, diese zwei Passiv-Sätze auf ihre Struktur hin untersucht und dann die Rollen für den ersten Satz entsprechend den Frame Elementen bestimmt. Der einzige Unterschied des FrameNet-Satzes ist, dass hier der Agent nicht bekannt ist und mit dem Default Agent „CNI“ versehen wurde. Dies stellt aber bezüglich des Gleichheitsvergleichs von syntaktischen Strukturen kein Problem dar, da die Abhängigkeiten die erforderlichen Informationen liefern.

Tabelle 23: Vergleich eines Passiv-Satzes mit einem FrameNet Passiv-Satz

nsubjpass(pressed-3, Wax-1) **auxpass**(pressed-3, was-2) **root**(ROOT-0, pressed-3) **prep**(pressed-3, into-4) pcomp(into-4, by-5) det(rod-7, a-6) pobj(by-5, rod-7)	**nsubjpass**(PRESSED-3, She-1) **auxpass**(PRESSED-3, was-2) **root**(ROOT-0, PRESSED-3) prt(PRESSED-3, back-4) **prep**(PRESSED-3, into-5) poss(seat-7, her-6) pobj(into-5, seat-7)

Neben der Abhängigkeit „prep()“, gibt es noch den „prt(Verb,y)“, die für Verben eine wichtige Rolle spielen. Das zweite Wort ist ein Partikel, das die Bedeutung des Verbs ändert. Beispiele für Partikel sind up, off, on, for usw. Zusätzlich dazu gibt es o.B.d.A. noch weitere Abhängigkeiten, die die Beziehungen zwischen jegliche im gleichen Satz vorkommenden Verben angeben. Diese sind advcl(), xcomp(), ccomp(), conj() und dep(). Außerdem ist normaler weiße in jeder root() ein Verb auf der rechten Seite. Angefangen mit dieser root() kann –muss aber nicht- eine Kette mit den anderen gerade genannten Abhängigkeiten eine Kette entstehen, die alle Verb-Verb-Beziehungen in einem Satz enthalten. Hat das Verb z.B. in root() mit den anderen Verben nichts zu tun, so sind nicht alle Verben in einer gemeinsamen Kette drin. Falls aber das root(ROOT-0,no_Verb) das Verb nicht enthält, so gibt es eine cop(no_Verb,Verb), woraus das Wurzel-Verb ermittelt werden kann. Alle anderen Verb-Abhängigkeiten, die in der Kette enthalten sind und angenommen mit dem root() verknüpft sind, enthalten nun Verben, die mit diesem „no_Verb“ in Kombination stehen können (z.B. advcl(no_Verb, Verb)). In jedem solchen Vorgang darf also die copulo-Abhängigkeit nicht außer Acht gelassen werden um mit dem richtigen Verb zu arbeiten. Copulo-Abhängigkeiten tauchen nur dann auf, falls nur ein Verb der Form „be“ im Satz auftaucht. Im Beispielsatz 4, der den folgenden Anfang besitzt “The cooling fan has to be controlled…”, steht im „root“ Knoten

das Verb „has“ , das mit dem Verb „controlled“ in Beziehung ist. Das „be“ dient nur als Hilfsverb für „controlled“. Somit wurde gezeigt, wie man anhand eines „main“-Verbs alle anderen abhängigen oder unabhängigen Verben einschließlich ihrer zugehörigen Nomen (nsubj) und anderen Zusätzen wie z.B. Präpositionen (prep) systematisch verwalten kann und für weitere Prozesse zur Verfügung stellt. Auf der anderen Seite wurde gleichzeitig die Kernidee des Kapitels 4.3.1 unterstützt.

4.3.5 Schlussfolgerung für zusammengehörige Namen (CN) und für die Wortartidentifizierung durch Auto SRL

Im vorherigen Kapitel wurden komplexe Strukturen mithilfe von FrameNet- und VerbNet-Muster verglichen um dabei den Satzgliedern Rollen zu verteilen. Dieser strukturelle Abgleich findet hier nicht statt. Es wird lediglich die Ontologie Ressource von WordNet eingesetzt, die für einzelne Tokens oder Token-Paare entscheidet ob sie ein zusammengehörendes Wort sind und wenn dann um welche Wortart es sich dabei handelt.

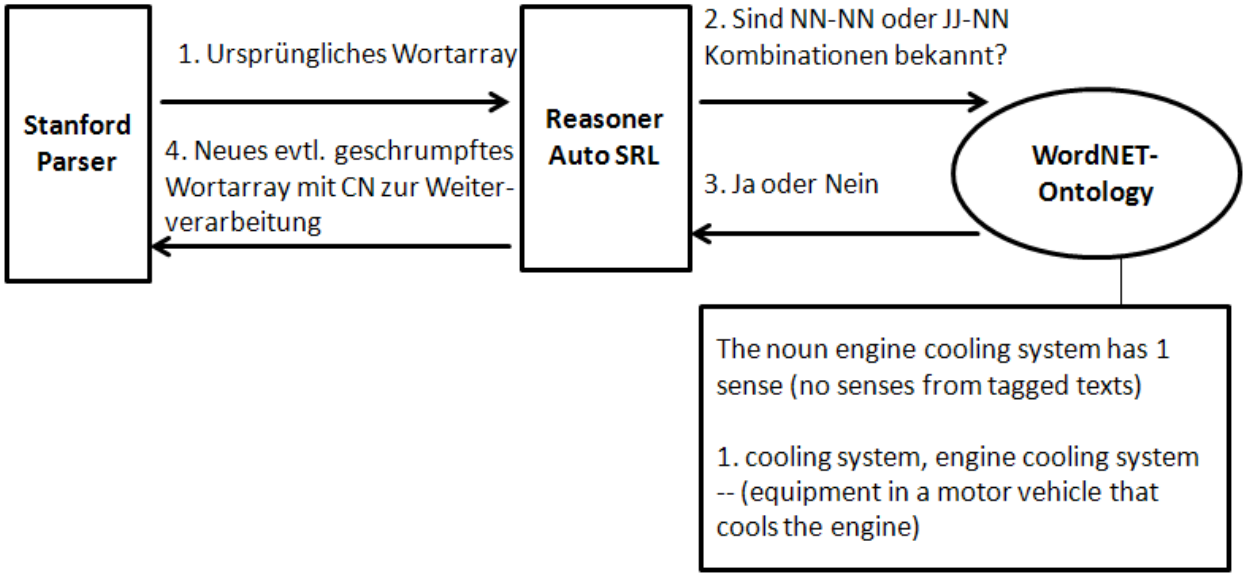

Abbildung 21: Schlussfolgerung durch Kombination von WordNet und Stanford

In den Grundlagen wurde erwähnt, dass in einem Text viele Namen vorkommen. Gleichzeitig kommen Namen öfters nebeneiander vor. Um diese zu erkennen hat z.B. AA den externen Name Entity Recognizer (NER) eingesetzt. Der NER gibt an, dass der gesuchte Name entweder ein Ort, eine Organisation oder eine Person sein kann. Beim Auto SRL wird diesbezüglich der WordNet in Kombination mit dem dieses Mal in Stanford integrierten NER eingesetzt. WordNet klassifiziert zwar Namen nicht untereinander, kann aber gleich feststellen, dass es sich um einen zusammengesetzten Namen handelt, falls dieses kombinierte Wort in der Datenbank liegt. WordNet weiß, dass sowohl „cooling system“ oder „enginge cooling system“ ein zusammengehöriger Nomen sind. Bevor Wörter vereinigt werden, sollten aber etwaige Probleme, die weiter oben festgestellt wurden, mit berücksichtigt werden. Betrachtet wird nun der Anfang des folgenden Satzes aus der Kategorie ‘Fans’ mit „The fans are controlled either

with a thermostatic switch or by the engine computer,…“. Nach der Anwendung der Vereinigung von Tokens sieht der Satz nun folgendermaßen aus: „The fans are controlled either with a thermostatic_switch or by the engine_computer. Man sieht hier, wie zwei nebeneinander stehende Nomen oder ein Adjektiv mit einem Nomen zusammengefasst werden können. Auch das Beispiel mit dem „cooling system“ zeigt, dass eine Verb-Nomen-Kombination genauso möglich ist, falls sie durch WordNet erkannt wird. Semantisch gesehen steht in diesem Fall nichts auf dem Weg diese Art von Token-Paaren zu kombinieren. Dagegen sieht es bei den erkannten Adverbien wie „a lot“ anders aus. Der Stanford Parser erkennt den Token „lot“ als Nomen. A_lot kann aber Stanford trotzdem nicht als Adverb erkennen. Daher werden Adverbien nicht verbunden, egal ob sie durch WordNet erkannt werden.

4.3.6 Vereinigung von thematischen Ontologie-Rollen

Eine Rollenzuweisung im Auto SRL erfolgt parallel zum Stanford durch VN und FN. Nachdem Stanford syntaktische Strukturen zur Verfügung gestellt hat, kann VerbNet anhand dieser Strukturen (genauer Verb-Strukturen) Rollen ermitteln. Wie schon weiter oben erwähnt gibt es Abhängigkeiten (advcl(), xcomp(), ccomp(), conj() oder dep(),…), woraus man syntaktische Verb-Ketten identifizieren kann. Entweder sind Verben voneinander unabhängig oder bilden so eine genannte Verb-Kette. Verb-Ketten tauchen vor allem bei konjunktionalen Nebensätzen öfters auf. Die nächste Tabelle zeigt ein Mapping für alle Frame Elemente aus dem Frame Cause_Motion. Das Verb „presses“ aus einem Beispielsatz ist als Lexical Unit auch in diesem Frame enthalten.

Tabelle 24: Mapping von Rollen im FN-Frame „Cause_Motion“

FN Rolle	**VN Rolle**
Agent	Agent
Cause	Agent, Cause
Theme	Theme, Instrument, Patient1
Source	Source, Patient2
Goal	Destination, Location
Area	Location

Auf der anderen Seite können auch FN Rollen für die Annotation von Rollen eingesetzt werden. Abhängig vom betrachteten Konzept (Frame) bietet FN interessantere Rollen bzw. Frame-Elemente als VN an. Als Beispiel schauen wir diesbezüglich den Satz „*A whois server listens [on tcp port 43] for requests from Whois Client*“ an. Für den gekennzeichneten Bereich gibt es im FN die Rolle Sought Entity, wobei VN diesen Bereich nur als Theme annotiert. Auf die

Rolle Theme kommt man entweder durch die VN-Klasse rummage-35.5 selbst oder durch die Role-Mapping-Datei zwischen FN und VN, drauf. Das Verb „listens“ besitzt das Frame (NP V PP.Location PP.Theme), das genau diesen obigen Zusammenhang mit den Tokens „listens“, „on“ und „for“ darstellt. Gleichzeitig gibt es im Stanford die Beziehung zwischen dem Verb und seinen Präpositionen, die diesen Zusammenhang folgendermaßen abbildet: prep(listens, on) und prep(listens,for).

Die nächste Tabelle stellt die Rollen des Frame „Seeking“ mit den Rollen der VN-Klassen gegenüber, wo auch das „listens“ sich befindet. Dabei ist zu achten, dass für das betrachtete Frame mehrere Klassen aus VN existieren können. Das gleiche gilt auch andersherum, wobei dies hier bei den unteren Tabellen nicht der Fall ist.

Tabelle 25: Mapping von Rollen im FN-Frame „Seeking“

FN Rolle	**VN Rolle**
Cognizer	Agent
Ground	Location
Sought_Entity	Theme

Aus semantischem Aspekt her sind FN-Rollen geeigneter als VN-Rollen. Auf der anderen Seite sind aber VN-Rollen praktischer, da ihre Strukturen und somit auch ihre Rollen tauglicher für Stanford sind. Es ist auch einfacher Stanford und VerbNet bezüglich der Annotation zu verknüpfen. Beide haben aber ihre Vor- und Nachteile. Aufgrund der engen Verbindung zum Stanford Parser soll VerbNet unbedingt bei der semantischen Annotation mit berücksichtigt werden. Durch Mapping-Möglichkeiten zwischen VN-Klassen und FN-Frames steht auch nichts auf dem Wege die reichliche semantische Information des FrameNet mit zu nutzen. Da manchmal mehrdeutige Klassen oder Frames für ein Verb existieren kann man hier die Schnittmenge beider Ontologien verwenden. Diese wird auch im Implementierungsteil ausführlich behandelt. Weitere Rollen sowie die Mappings zwischen ihnen sind entweder im Anhang oder auf den Herstellerseiten (VerbNet) teilweise angegeben.

Abschließend wird noch gezeigt, welchen Vorteil der Einsatz dieser drei Ontologien neben dem Stanford Parser bezüglich Rollengewinnung mit sich bringt. Die Anforderung „The traffic lights allow pedestrians to cross the road safely” wird zuerst nur mit Hilfe des Parsers semantisch annotiert. Man sieht in der Abbildung 22, dass den Konstituenten nur Default-Rollen des SALEs zugewiesen wurden. In der Abbildung 23 wurde zur Veranschaulichung ein logischer Filter eingesetzt. Dies soll darauf hin weißen, dass eine effektive Rollengewinnung nur durch Schlussfolgerung der Teilergebnisse beider Ontologien erfolgen kann.

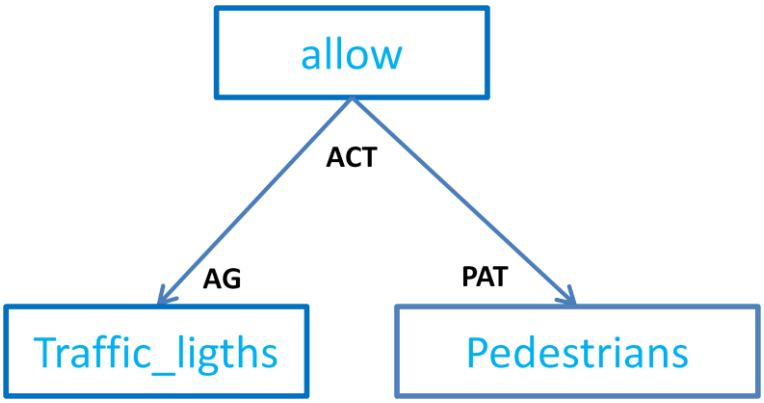

Abbildung 22: Durch Stanford Parser ableitbare Beziehungen

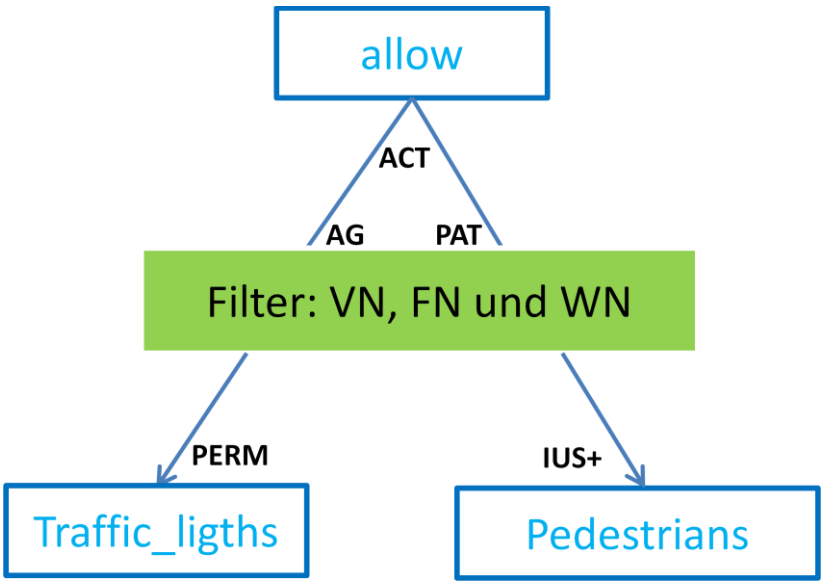

Abbildung 23: Erweiterte Beziehungen durch Hinzunahme von VerbNet, FrameNet und WordNet

4.4 Zusammenfassung

Das Hauptaugenmerk in diesem Kapitel war die Erkennung von vollständigen Teilsätzen sowie das Beachten von kleinsten Satzbestandteilen wie Nomen, Adverbien, Adjektiven und vor allem von Verben. Ein Teilsatz wurde in Kombination mit einem oder mehreren Verben näher betrachtet. Der Startpunkt war also das Verb-Spektrum. Alle Rollen der zum Verb abhängigen Satzglieder, wurden bei der semantischen Annotation durch Ontologien vergeben. Im nächsten Kapitel werden die hier entwickelten Konzepte teilweise bezüglich ihrer Implementierung erläutert. Das Gesamtkonzept des Auto SRL besteht aus einer Kombination sowie einer Ergänzung von WordNet, Stanford Parser, VerbNet und FrameNet.

5 Prototyp für „Auto SRL“

Nicht in jeder Forschungsarbeit muss massiv mit Ontologien gearbeitet werden, wie in solchen. Zum Prototyp integrierte Ontologien wurden fast alle als API integriert. Da Erweiterungen direkt an den APIs durchgeführt wurden, folgerten sich hierdurch die Namen der Haupt-Packages (WordNetAPI, VerbNetAPI …). Zunächst wird grob gezeigt, welche Komponenten es gibt und in welcher Aufgabe sie eine Rolle spielen.

5.1 Grobes Vorgehen bei der Informationsextraktion und dabei integrierte Komponenten

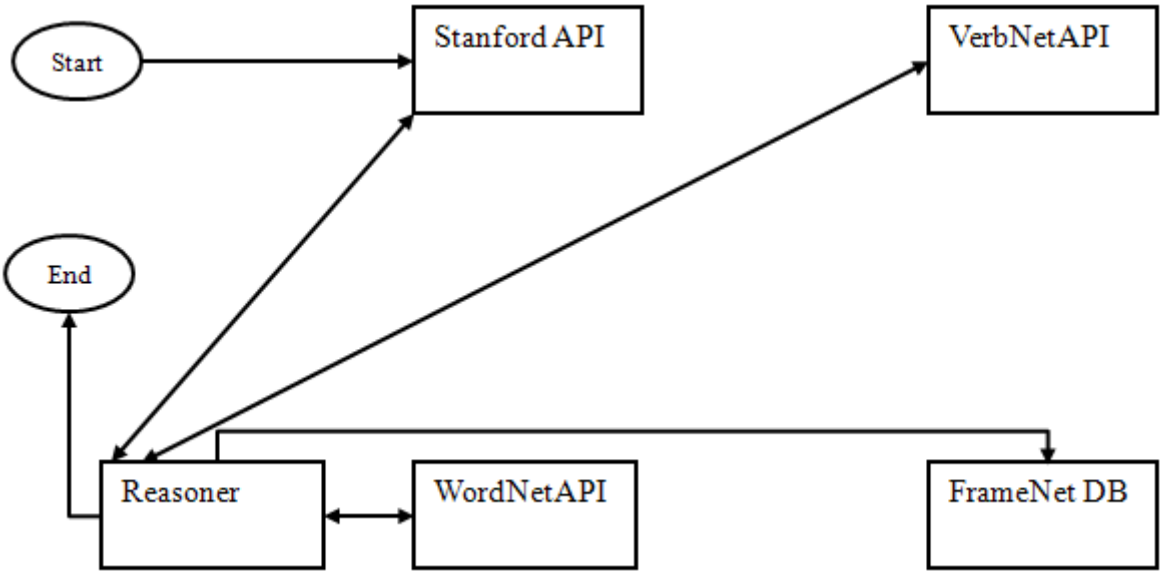

Abbildung 24: **Auto SRL** - Eine Übersicht ohne GUI

1. Ermittlung von Referenzen durch Stanford Core NLP (vor der Wortverschmelzung)
2. Ermittlung der Wortart und Erkennung von speziellen Namen über das Teilprogramm NER des Stanford Core NLP
3. WordNet Abfrage über passende Wortkombinationen und der Versuch einer Korrektur der Ergebnisse aus vorherigem Schritt
4. Ermittlung von potentiellen Verben durch VerbNet und Stanford Strukturen
5. Ermittlung der Abhängigkeiten (Dependencies), Baumstrukturen (Trees) und Wortarten (POS) durch den Stanford Core NLP
6. Extraktion von Semantiken mit Hilfe von VerbNet und FrameNet
7. Ermittlung der Referenzen durch das Teilprogramm Coref des Stanford Core NLP

8. Schlussfolgerung durch Reasoner

Alle vier Komponenten sind die sogenannten Informationsbeschaffer. Das gesamte System baut auf diese auf. Einige Teile dieser Komponenten werden mehrmals aufgerufen, sodass das Gesamtprogramm nicht wie hier dargestellt seriell abläuft. Die zusätzliche Arbeit übernimmt der Reasoner (Schlussfolgerer), der die hier gewonnenen Informationen zusammenfügt und logische Schlüsse daraus zieht.

5.2 Ergänzungen zum Stanford Core NLP

Stanford API liefert für jeden Satz grundsätzlich die Abhängigkeiten, die Baumstruktur und die Wortarten. Die Hintergrundidee hinsichtlich der Erweiterung des Stanford im Prototyp ist es eigentlich die gemeinsame Nutzung der Abhängigkeiten und der Baumstruktur eines Satzes zu ermöglichen um dabei mit dem „reasoner“ neue Erkenntnisse zu folgern. Die Methode „*PhrasalhasImportantLeaf*“ gibt an, dass die betrachtete Phrase mindestens ein Blatt besitzt, das das für die Semantik eine wichtige Rolle spielt. Da es in einem Satzteil mehr als nur ein Verb existieren kann und somit mehrere semantische Informationen gefolgert werden können, werden im gleichen Baum unterschiedliche Strukturen mit der Klasse „reasoner.*pickSemanticTreeStructure*„ „reasoner.InformationaboutVerb“ vom Baum mit den VerbNet-Frames verglichen. Dabei wird Stanford.Dependency Klasse aufgerufen, damit auf die richtigen Stellen des Baumes verwiesen wird.

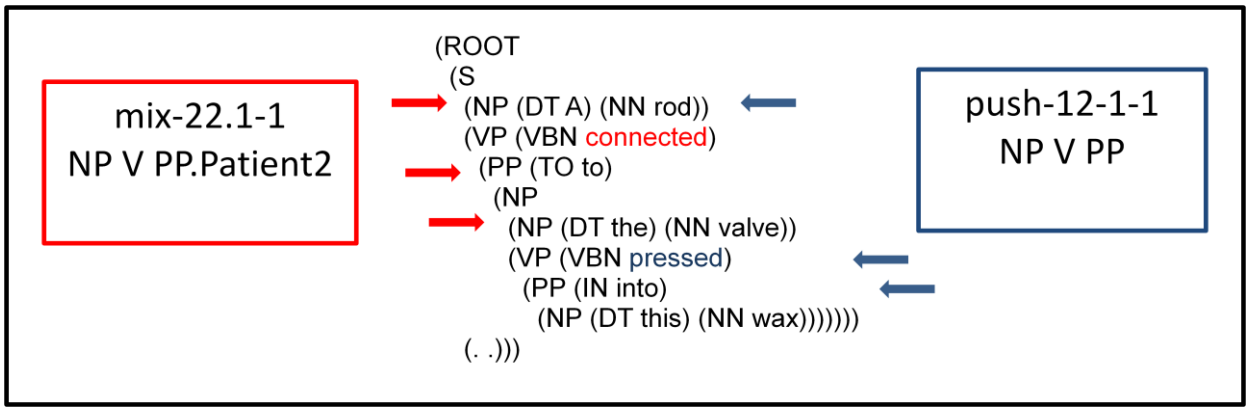

Abbildung 25: Entscheidung des Reasoners bei potentiellen Verben

Nach welchen Kriterien hier nach einem Verb geschaut wird oder wann es sich um ein potentielles Verb handelt, das der Parser nicht erkennt, werden in den nächsten Kapiteln gezeigt.

5.3 WordNet als unterstützender Faktor der Kernstrategie

Zusammen mit den Stanford-Strukturen entwickelte Lösungsstrategie braucht auch die Unterstützung aus der WodNet-Seite. Neben der Klasse Wordnet.*CNthroughWordNet(WordArray)*, die zusammengehörige Namen durch WordNet-DB ermittelt, gibt es eine weitere Klasse namens WordNet.*maybeVerb(Verb)*, die ermittelt, dass es sich um ein Verb handeln könnte. Der Reasoner ermittelt anhand dieser und weiterer

Informationen z.B. durch VerbNet, dass das Verb wirklich ein Verb ist, das z.B. durch den Parser nicht erkannt werden konnte. Hierbei dient WordNet als kontrollierender Faktor.

Die Zusammenarbeit zwischen Stanford, WordNet und VerbNet erfolgt folgendermaßen:

1. Durch WordNet lässt sich feststellen ob potentielle Verben im Satz sind, die durch Stanford nicht als solche erkannt wurden
2. Ermittlung der Verb-Frame Paare, damit dadurch ein Vergleich mit Stanford durchgeführt werden kann
3. Jedes einzelne VN-Frame wird nun mit der ermittelten Stanford Baum-Struktur einzeln verglichen. Das potentielle Verb wird aufgrund des Vergleichs in Präteritum umgewandelt. Falls Strukturen passen, so können aus dem neuen Verb heraus Semantiken extrahiert werden

Dieser Vorgang wird nur dann durchgeführt, falls (potentielle) Verben nach der Umwandlung in das Präteritum mit dem root-Element (z.B. Verb) des Baumes kettenförmig nicht gebunden sind oder anders ausgedrückt keine Abhängigkeiten zwischen den betrachteten Verben bestehen. Falls eine Abhängigkeit bestehen würde, so wäre es einfacher die Rückschlüsse – ein Verb besitzt ein Subjekt oder ein Objekt - für ein Verb über das jeweilige root-Element zu ziehen. Im Kapitel 4 wurden für die Erkennung der abhängigen Verb-Strukturen die Dependencies „xcomp()“, „advcl()“ usw. eingesetzt.

5.4 VerbNet als Vermittler zwischen Parser und FrameNet

In der VebNetAPI gibt es drei Klassen, die die semantisch gefüllten XML-Files bearbeiten sollen. Zurzeit gib es in der Verbnet-DB mit der Version 3.1 270 XML-Files, die Klassen und Unterklassen enthalten. Die Unterklassen erben dabei alle Eigenschaften der oberen Klassen. Ein Frame einer oberen Klasse wird zwar nach unten zu den Kinderklassen übertragen aber nicht anders herum. Deswegen ist es immer so, dass die Unterklassen mehr Frames besitzen können als die oberen Klassen. Die jeweiligen Informationen (VNCLASS-ID, Member, Themrole, FRAMES, SYNTAX, SEMANTICS, …) in den Tags dieser XML-Dateien können durch die w3c.dom Bibliotheken iteriert werden.

Die Java Klasse *Inspector*, die auch die Main-Klasse des APIs ist, ruft die *performInspection()*Methode auf. Diese Methode ruft wiederum innerhalb einer Schleife 1 bis max. 270 (Länge aller XML-Dateien) mal die Methode *processNode(class)* in der gleichen Klasse auf. Dieser Aufruf stellt aber einen Engpass dar, da jede Datei einzeln geöffnet, bearbeitet und wieder geschlossen werden muss, die erheblich Zeit in Anspruch nimmt. Aufgrund des Geschwindigkeit-Problems wurde das *VerbNetAPI* mit der Klasse *VNClass* erweitert. Diese Klasse verwendet einen vorhandenen Index, der in einer einzigen Datei *vn-fn.xml* abgelegt ist. Bei dieser Datei handelt es sich um eine Mapping-Datei (siehe Abbildung 26), die vom Unified Verb Index (UVI bzw. VerbNet) für die Kombination dieser zwei (von vier) Ontologien zur Verfügung gestellt wurde. Anhand dieser Mapping-Datei ist aber eine

Suche in den XML-Files vom VerbNet nicht ganz optimal gewesen, woraufhin im Rahmen dieser Arbeit Namen der XML-Files händisch (in Abstimmung mit der Projektleiterin von VerbNet) angepasst wurden. Somit war es anhand des Verbs möglich, die wirklich relevanten XML-Datei(en) bzw. Klassen und Unterklassen zu bearbeiten. Die Methode *VNClass.getVNClass* bekommt als Eingabe ein Verb, das im VerbNet als Mitglied (member) bezeichnet wird. Anhand dieses Verbs ermittelt die *getVNClass* Methode den zugehörigen Namen der Klasse, der gleichzeitig die Bezeichnung für die XML-Datei ist. Das Hauptziel vom UVI gleichzeitig von VerbNet ist es mehrere Ontologien gemeinsam zugreifbar zu machen, wovon in dieser Arbeit bei der Kombination von Verb- und Frame-Net durch eine Anpassung (wie gerade erwähnt) profitiert wurde.

Nach dem die Funktionsweise vom VerbNet API teilweise erläutert wurde, kommen jetzt Hinweise über die Aufrufparameter, die über *run* (bzw. *main)* der *Inspector*-Klasse dem API übergeben wurden, damit nach Verben und Frame-Strukturen korrekt gesucht wird. Bei den API-Javadocs gibt es eine Komplett-Liste über die Parameter oder die sogenannten Operatoren, die beim Aufruf von *Inspector* verwendet werden können. Die bezüglich des Prototyps verwendeten Operatoren sind *–V* und *–O*. Mit *–Vq* gibt der *Inspector* alle gewünschten Verb-Struktur Paare mithilfe von *Sweeper.java* aus. Die aufgerufene Methode vom Inspector (hier *run*) liefert einen String zurück, sodass diese den Inhalt der Verb-Frame Paare für die Weiterverarbeitung durch Auto SRL zur Verfügung stellt. Mit dem anderen Operator *–O Dateiname* kann man den Inspector erzwingen, dass er nur bestimmte Dateien untersucht. Die vorhin erwähnte Zeitgewinnung durch Anpassung der Mapping-Datei wird zusammen mit diesem Operator realisiert.

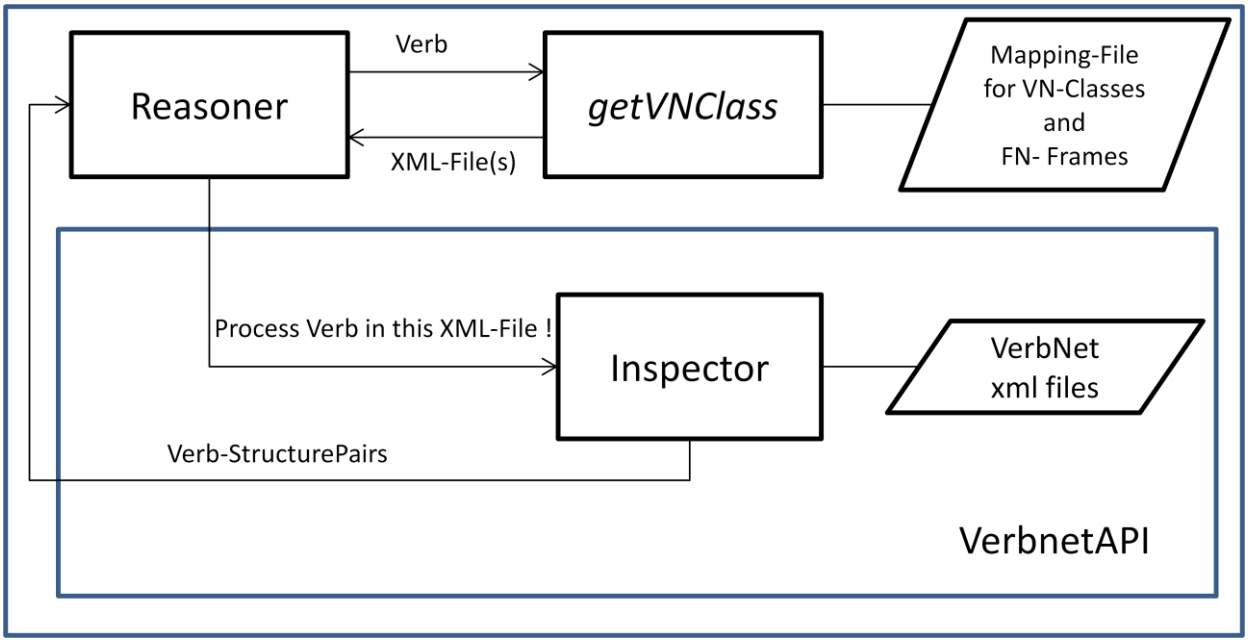

Abbildung 26: Zeitgewinnung im VN-API durch vorherige Klasseneinschränkung

Aufgrund der Mehrdeutigkeit der Struktur-Ergebnisse im VerbNet wurden noch andere Abhängigkeiten vom Stanford eingeschaltet, damit die Klassen-Frame-Suche optimiert werden

konnte. Je besser hier die Ausbeute ist, desto optimierter wird nachher das Mapping zwischen Verb- und Frame-Net bzw. dann später zwischen diesen zwei und den anderen Ontologien wie z.B. PropBank, OntoNotes Sense oder OpenCyg.

5.5 FrameNet als unterstützender Faktor für Rollenvergabe

Die Rollenvergabe durch FrameNet hat andere Dimensionen, wie der gerade vorgestellte VerbNet, da hier nicht nur das Verb, sondern auch Substantivierte Verben sowie Adjektive im Visier des „semantischen Annotators" liegt. Außerdem bietet FrameNet bessere Übersicht bei der Aufteilung der Frame Elemente (Rollen).

Zunächst wird geschaut, zu welchen Frames der Beispielsatz aus Anhang D passen würde. Bei der Annotation des durch Stanford falsch interpretierten Satzes wurde festgestellt, dass FrameNet den gleichen Fehler nicht gemacht hat, da „presses“ im FrameNet als Verb richtig erkannt wurde.

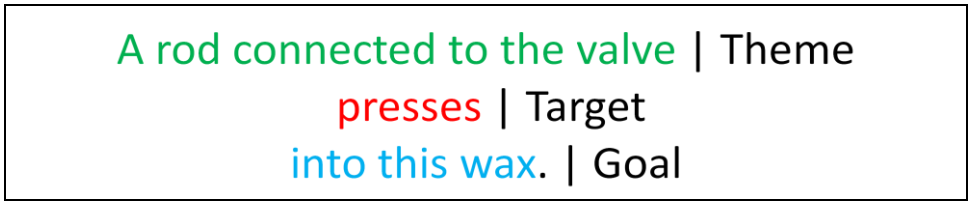

Abbildung 27: FrameNet-Frame: Cause_motion

In der nächsten Abbildung wird gezeigt, in welchem Frame ein anderer Teilsatz dieses Satzes liegt. Dieses Mal wird das Verb „connected“ betrachtet. Die abhängigen Frame-Elemente zu diesem L.U. sind „a rod“ und „to the valve“, die entsprechend der Abbildung 28 annotiert wurden.

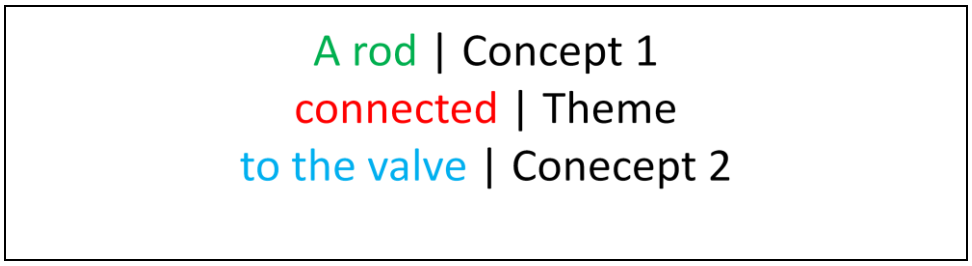

Abbildung 28: FrameNet-Frame: Cognitive_Connection

Das FrameNet API Semafor [DSCS10] braucht 8 GB RAM um sogar einen einzigen Satz zu annotieren. Der Grund hierfür ist die Abhängigkeit zum MST Parser, der das große durch „English Tree Bank“ trainierte Modell zuerst laden muss. Dieses Problem wollte man hier umgehen. Diese Umgehung danken wir aber dem Stanford Parser, der auch mit „English Tree Bank“ zu tun hat. Die Entscheidung welche Frames letztendlich aufgrund ihrer syntaktischen Struktur untersucht werden sollten, wird neben den Parser-Baumstrukturen (NP, VP, TO…) zusätzlich durch die Mapping-Datei und durch das VerbNet beeinflusst.

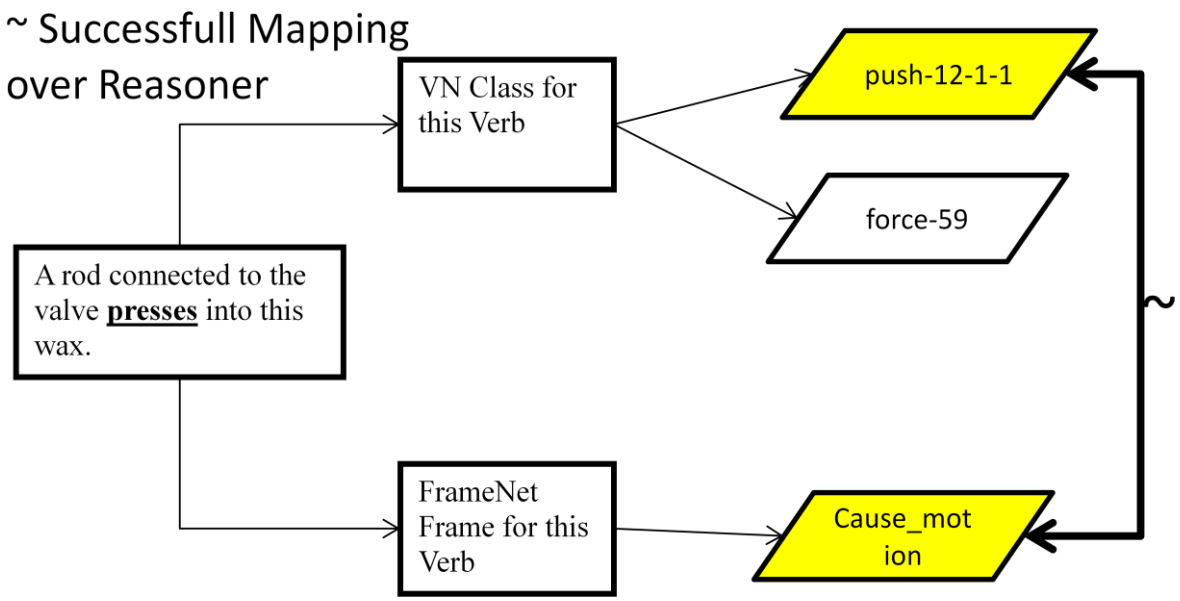

Abbildung 29: Wahl der Frames anhand von Mapping-Datei und VerbNet-Strukturen

In der Abbildung 29 sieht man, wie sich VN und FN gegenseitig unterstützen. Dabei wird anhand der aus vorigem Kapitel bekannten Mapping-Tabelle geschaut, zu welchen VN-Klassen und FN-Frames das Verb passt. Es entstehen manchmal mehrdeutige Ergebnisse. Passt eines der Ergebnisse (Klassen) vom VN zu einem Ergebnis (Frame) vom FN, so kann man dieses Frame-Klassen-Paar im Reasoner-Packet als passend kennzeichnen. Reasoner vergibt dann die entsprechenden Rollen aufgrund des erfolgreichen Mappings. In beiden Ontologien bekommt man öfters mehrdeutige Ergebnisse. Aber sobald man zu einem Ergebnis sein Gegengewicht in der anderen Ontologie findet, so kann man Mehrdeutigkeiten beseitigen. Falls im FrameNet-Frame keine syntaktischen Einträge z.B. für „press.v“ existieren, so wird dieses Paar eliminiert und nicht mehr berücksichtigt. Zur endgültigen Rollenvergabe wird das übrig gebliebene passende Paar herangezogen. Weitere Mehrdeutigkeiten können durch weitere Abhängigkeiten zusammen mit weiteren Beispielsätzen dieser Ontologien eliminiert werden. Die Realisierung des letzten Schrittes, worauf auch im Kapitel 6.2 hingewiesen wird, könnte in weiteren Arbeiten durchgeführt werden.

5.6 Ausgaben des Auto SRL

Als aller erstes wird, außer der Referenzauflösung, das Teilprogramm NER auf den Satz- bzw. WortArray angewendet. Dabei schrumpft das ursprüngliche WortArray. Der ähnliche Ablauf, der mit WordNet stattfindet, wird in der nächsten Abbildung veranschaulicht.

In der Abbildung 21 (Kapitel 4.3.5) sieht man wie über WordNet ein spezielles zusammengesetztes Word sofort ermittelt wird. Sobald so ein Word erkannt wird und es ein Name ist, können deren Teilwörter zusammengebunden werden. Dieser Vorgang verhindert dann auch Fehler in den nächsten Schritten. Stanford Parser erkannt dann von WordNet ermittelten zusammengesetzte Wörter als Namen. Der mehrmals analysierte Satz „A rod

connected 63ot he valve presses into this wax.“ Soll dabei auch teilweise zur Veranschaulichung der Zwischenergebnisse dienen. Weitere Beispiele können, wenn nötig, auch herangezogen werden.

5.6.1 Ermittlung von zusammengesetzten Wörtern

In der WRSPM-Umgebung wurden aus formalen Gründen zusammengehörige Wörter wie z.B. „traffic_lights“ mit einem verbindenden Zeichen zusammengefasst. Dieses Vorgehen unterstützt einerseits der Parser mit seiner Abhängigkeit (nn) und andererseits der WordNet durch seinen Datenbankeintrag. Das WordNet trägt dazu bei, dass auch Adjektiv-Nomen-Paare (z.B. „thermostatic switch“, Beispielsatz aus 5.6.3) kombiniert werden können. Zusätzlich zu dem wurde beim Stanford sicherheitshalber der NER auch für diesen Zweck eingeschaltet.

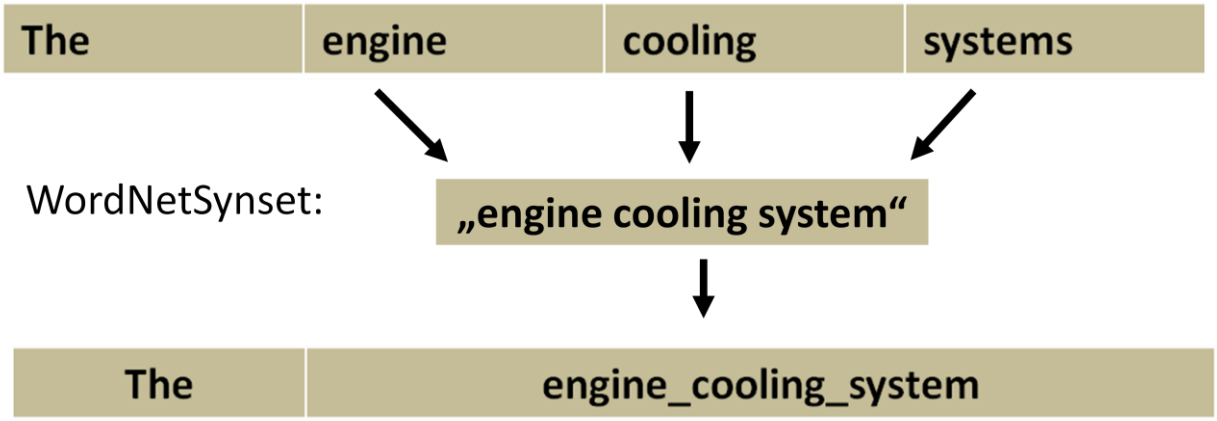

Abbildung 30:WordNet: Vereinigung von Tokens

Der gewählte Satz enthält keine Wörter solcher Art, weshalb diese Tokens genutzt wurden. Aus ursprünglich vier Satzbestandteilen existieren nur noch zwei. Für den neu entstandenen Token (engine_cooling_system) weist der Stanford Parser direkt die Wortart Nomen zu.

5.6.2 Ableitung von Ontologie-Rollen durch Ermittlung des Verbs und seinen Abhängigkeiten

Über WordNet lässt sich schnell ermitteln, dass ein Wort zu mehreren Kategorien (Nomen, Verben, Adverbien, Adjektive) gehören kann. Das Wort „press“ in Kombination mit „into“ ist auf der anderen Seite ein Hinweis für Auto SRL, dass es sich bei diesem Token um ein Verb handelt. Dies Problem löst Auto SRL grundsätzlich über FN, wo der lexikalische Eintrag zum Verb „press“ angibt, dass „press“ eine Präposition mit „into“ enthält. Der letzte Grund verstärkt auch die Annahme, womit „presses“ nun mit „pressed“ ersetzt wird, damit der Parser sie das nächste Mal als Verb erkannt.

Als Rollen kommen hier aufgrund der VerbNet-Klasse (push-12-1-1) „Agent“ und „Theme“ in Frage. Da aber FrameNet etwas präziser mit den Rollen ist, werden durch die Unterstützung von Abhängigkeiten nur noch FN-Rollen den Konstituenten des Satzes zugeordnet. Die unteren

Abhängigkeiten weisen z.B. auf die Rolle „Goal“ des FN hin. Parallel dazu ist auch im FN der lexikalische Eintrag „PP[into].Dep“ in der Rolle „Goal“ hinterlegt. Die farbigen Rollen in der unteren Annotation deuten auf das FN-Frame „Cause_motion“ hin. Diese Rollen werden auch durch Abhängigkeiten bekräftigt. Leider kann hierdurch der Agent von „pressed“ immer noch nicht ermittelt werden. Es liegt auf jeden Fall links von „pressed“, da hier der Theme-Bereich sich befindet. Mit der prep()-Abhängigkeit kann immer festgestellt werden, dass der linke Teil von „prep()“ ein Verb enthält. Hierdurch bekommt „pressed“ die Rolle ACT. Eine weitere Abhängigkeit „partmod(valve-6, pressed-7)“ gibt der Parser auch aus. Diese Abhängigkeit ist leider nicht korrekt, die in Kombination mit FN in der nächsten Annotation korrigiert werden soll.

#A #rod #connected #to #the #valve #presses #into #this #wax.

[#A rod connected #to #the valve$_{\text{Theme}}$ pressed |ACT$_{\text{Target}}$ #into #this wax|PAT$_{\text{Goal}}$].

prep (pressed-7 , into-8) det (wax-10 , this-9) pobj (into-8 , wax-10) → PAT

prep(pressed-7, into-8) → ACT

Neben „pressed“ gibt es ein anderes Verb „connected“, der eventuell auf ein Subjekt hinweist, da „Connected“ sich im Bereich von „Theme“ befindet. Falls also ein Subjekt für „connected“ existiert, so ist es auch gleichzeitig ein indirektes Subjekt des obigen Verbs. Der Vorgang zeigt, wie der indirekte Agent (Subjekt) für „presses“ aus dem Theme-Bereich ausgepickt wird.

#A #rod #connected #to #the #valve #presses #into #this #wax.

[#A rod|AG$_{\text{Concept1}}$ connected|ACT$_{\text{Target}}$ #to #the valve|PAT$_{\text{Concept2}}$ pressed into this wax].

nsubj(connected-3, rod-2) →AG

prep (connected-3, to-4) det(valve-6, the-5) pobj(to-4, valve-6) → PAT

prep(connected-3, to-4) oder nsubj(connected-3, rod-2) → ACT

Sobald man anstelle von einigen unsicheren Abhängigkeiten wie „partmod“ vorhandene Ontologie-Informationen nutzt, so ist man o.B.d.A. auf der sicheren Seite. Die Abhängigkeiten werden nach der Zuweisung der Ontologie-Rollen wieder genutzt, damit auch die SALE Rollen innerhalb der Frame-Elemente (farbige Hervorhebungen) integriert werden können.

Insgesamt wurde hierdurch gezeigt, wie wichtig VerbNet für beide Richtungen ist. Auf der einen Seite kann man zwischen ihm und Stanford Daten vergleichen und auf der anderen Seite kann man über ihm an andere Informationen vom FrameNet kommen. Eine Mapping-Möglichkeit zwischen SALE und VN scheint realistischer zu sein, da in beiden Konzepten auf Konstituenten-Basis – mit der Ausnahme von Satz-Rollen – Rollen vergeben werden. Dagegen

sind FN-Rollen nicht immer auf eine einzige Konstituente beschränkt. Die farbigen Hervorhebungen sind nicht Bestandteil des Prototyps. Farben wurden aber übersichtlichkeitshalber eingeprägt.

5.6.3 Auflösung von Referenzen

Die „coref"-Eigenschaft von Stanford, hilft entweder gleiche Nomen zueinander zuzuordnen oder Referenzen aufzulösen. Diesbezüglich wird ein anderer Satz untersucht. Obwohl er ein langer Satz ist, werden Referenzen korrekt aufgelöst. Die SALE-Rollen EGD und EGQ deuten darauf, dass die linken Seiten mit den rechten Seiten (z.B. They durch Fans) ersetzt werden sollen. Somit schafft man eine Ordnung zwischen den Nomen und ihren Referenzen.

#The #fans #are ….. #the #temperature #drops #below #that #point.

The fans are controlled either with a thermostatic switch or by the engine computer, and they turn on when the temperature of the coolant goes above a set_point|Ground. They turn back off when the temperature drops below that point.

[@point|EQD @point |EQK]. [@temperature|EQD @temperature |EQK].

[@They|EQD @Fans |EQK]. [@They|EQD @Fans |EQK].

Bei der Zuordnung von VN- und FN- Rollen wird für jedes unterstrichene Verb analog wie in den ersten Beispielen vorgegangen. Beim ersten Teil des ersten Satzes ist zu beachten, dass es sich um einen Passivsatz handelt. Daher bekommen „thermostatic_switch" und „engine_computer" die Rolle des Subjekts (AG), wobei „fans" die Rolle des Objekts (PAT) bekommt. Dies alles ist die Aufgabe des Reasoners, der mit einer Bedingungsabfrage (Passiv oder Aktiv?) dieses Problem umgeht. Die Konjunktion „or" im gleichen Satzteil wurde durch Reasoner nicht wirklich berücksichtigt, da das Verb „controlled" bezüglich der Abhängigkeiten und der Baumstruktur seinen Einfluss bis zum Satzzeichen (,) besitzt. Dagegen wird der zweite Teil des ersten Satzes in zwei Bereiche aufgeteilt. Der erste Teil liegt zwischen dem „S" (nach dem „and") und dem „SBAR", wobei letzteres die Grenze des „when"-Teils darstellt. Beide Faktoren (Grenzen sowie Verbstrukturen) spielen also für die Analyse des Satzes eine wichtige Rolle.

6 Ausblick

6.1 Analyse von weiteren Satz-Gerüsten

Verb- und Frame-Net Wissen wurden anhand von Schnittmengenbildung in der SRL der Texte eingesetzt. Dabei lag die Fokussierung auf den Verb-Frame Paaren des Verb-Net sowie auf der lexikalischen Einheit als Verb im Frame-Net. Die anderen lexikalischen Einheiten (Adjektive sowie Substantivierte Verben) könnten noch in diesem Prozess einbezogen werden; inwieweit diese Informationen für SRL hilfreich sind, könnten noch untersucht werden.

6.2 Vergleich von Sätzen mit Ontologie-Sätzen

In den hier erwähnten Ontologien gibt es genügend Muster-Beispielsätze, die mit dem zu untersuchenden Satz verglichen werden könnten. Im Kapitel 4.3.4 existiert solch ein Vorgang, der in der Abbildung 20 sowie in der Tabelle 23 zu finden ist. Dabei sollten bestimmte Abhängigkeiten des Stanford von beiden Sätzen miteinander verglichen werden. Angenommen der untersuchte Satz hat eine Abhängigkeit „poss()“, dann sollte auch der Ontologie-Satz die gleiche Abhängigkeit besitzen.

6.3 Effektivere Nutzung des Stanford Parsers bei SRL

Die Wichtigkeit des Stanford Parsers bezüglich der Syntax ist mittlerweile bekannt. Doch was muss noch gemacht werden, damit die hier gewonnene Information besser in SRL nutzbar ist. Diesbezüglich können z.B. noch mehr Abhängigkeiten untersucht werden, die die Verbindung zwischen der Syntax und der Semantik erleichtern.

6.4 Standardisierung von thematischen Rollen

In dieser Arbeit wurde versucht, die SALE-Rollen, die sich im Anhang E befinden, den Konstituenten des Textes zu zuordnen. In manchen Fällen (siehe Anhang H) konnten keine SALE-Rollen für die Rollen des FrameNet sowie VerbNet ermittelt werden. Daher sollten alle thematischen Rollen entweder in einem Standardisierungsprozess vereinheitlicht werden oder falls dies nicht möglich ist anhand von Mappings zueinander abbildbar sein. Solch eine Vereinheitlichung könnte eine noch breitere Akzeptanz bei den Anwendern finden.

Literaturverzeichnis

[Abb83] ABBOTT, Russell J.: Program Design by Informal English Descriptions.In: *Communnications of the ACM* 26 (1983), November, Nr. 11, S. 882–894. DOI 10.1145/182.358441. – ISSN 0001–0782

[AEW08] J. Angele, M. Erdmann and D. Wenke, Ontology-Based Knowledge Management in Automotive Engineering Scenarios. In: Ontology Management, Vol. 7 Springer, ISBN 978-0-387-69900-4, 2008

[ArZh04] M. Aref, Z. Zhou, The Ontology Web Language (OWL) for a Multi-Agent Understanding System, Department of Computer Science and Engineering, Bridgeport, 2004

[Bern00] T. Berners-Lee: Semantic Web on XML, W3C, http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html, letzter Zugriff am 22. August 2011

[Dai97] L. Daigle, WHOIS Protocol Specification. The Internet Engineering Task Force, RFC3912, 2004

[Den07] O. Denninger, Erweiterungen des Kantenkonzepts deklarativer Grafersetzungssysteme von Einfachkanten über Hyperkanten zu „Superkanten“, Diplomarbeit, Institut für Programmstrukturen und Datenorganisation, Karlsruhe, 2007

[Dir11] Directgov, Public Services all in on way. http://www.direct.gov.uk/en/TravelAndTransport/Highwaycode/DG_070108, letzter Zugriff 15.01.11

[DSCS10] Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith, Probabilistic Frame-Semantic Parsing in Proceedings of NAACL-HLT, 2010

[ENZ04] D. Estival, C. Nowak und A. Zschorn, Towards Ontology-Based Natural Language Processing, Human System Integration Group, Edinburgh, 2004

[ENZ04] D. Estival, C. Nowak und A. Zschorn, Towards Ontology-Based Natural Language Processing, Human System Integration Group, Edinburgh, 2004

[Fürn10] J. Fürnkranz, Einführung in Data und Knowledge Engineering, The Semantic Web, TU-Darmstadt, 2010.

[Gali11] Java Break Iterator http://openbook.galileodesign.de/javainsel5/javainsel04_004.htm#bild

[Gelh01] T. Gelhausen, Topic Maps und ihre Anwendbarkeit im webbasierten Lern- und Autorensystem Companion, Studienarbeit Fakultät für Informatik, Universität Stuttgart, 2010

[Gelh10] T. Gelhausen, Modellextraktion aus natürlichen Sprachen – Eine Methode zur systematischen Erstellung von Domänenmodellen, KIT Scientific Publishing, ISBN 978-3-86644-547-5, Karlsruhe, 2010.

[Gene91] M. Geneserth, Knowledge Interchange Format, In Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning, Allen, 238-249, 1991

[Heim06] Dr. M. P. E. Heimdahl, Requierements what and how?, UMSEC, Software Engineering Center, University of Minesota, 2006

[JaLa11] M. Jastram, L. Ladenberger, Nachverfolgbarkeit zwischen formalen Modellen und Anforderungen in natürlicher Sprache, Heinrich Heine Universität, Düsseldorf, 2011.

[Jast10] M. Jastram, ProR-Eine Softwareplattform für Requirements Engineering, Heinrich Heine Universität Düsseldorf, 2010.

[JHLJ10] M. Jastram, S. Hallerstede, M. Leuschel, A. G. R. Jr, An Approach of Requirements Tracing in Formal Refinement, VSTTE, Edinburgh, 2010.

[Kiss04] L. Kissinger, Ortspräpositionen und Richtungspräpositionen, http://www.lothar-kissinger.de/grammar/prepositions.htm, letzter Zugriff 18.01.12

[KIT11a] KIT, Karlsruhe Institut of Technology, Institute for Program Structures and Data Organization. https://svn.ipd.uni-karlsruhe.de/trac/mx/wiki/RECAA/Tools/AutoAnnotator. Zuletzt besucht am 29. Januar 2012

[KIT11b] KIT, Karlsruhe Institut of Technology, Institute for Program Structures and Data Organization, SVN-Downloads, for eSale, AutoAnnotator … http://svn.ipd.uni-karlsruhe.de/repos/koerner/mx/mx_update_site/site.xml, letzter Zugriff am 25. Januar 2012

[KLLS06] C. Kunze, L. Lemnitzer, H. Lüngen, A. Storrer, Modellierung und Integration von Wordnetzen und Domänenontologien in OWL am Beispiel von GermaNet und TermNet, Heidelberg, 2006

[Land10] M. Landhäußer, Automatische Auszeichnung von Semantik in natürlichsprachlichen Spezifikationen, IPD Karlsruhe, 2010

[Mitk03] R. Mitkow, The Oxford Handbook of Computational Linguistics, Oxford University Press, ISBN 0-19-823882-7, Oxford, 2003

[MSM04] M. Marcus, Beatrice Santorini and M.A. Marcinkiewicz, Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics*, volume 19, number 2, pp313-330.

[NiPe01] N. I., A. Pease, Toward a Standard Upper Ontology, in Proceedings of the 2nd International Conference on Formal Ontology in Information Systems, 2001

[NiPe03] I. Niles , A. Pease, Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology, Teknowledge, Palo Alto, 2003

[Onto09] http://62.204.194.27/omnigator/models/topic_complete.jsp?tm=topicmap.xtm&id=tosca , Ontopia, 2009

[Palm11] M. Palmer, A class based Verb-Lexicon, Department of Linguistics, University of Colorado Boulder, http://verbs.colorado.edu/~mpalmer/projects/verbnet.html, letzter Zugriff 22. August 2011

[Peas09] A.Pease, Standard Upper Ontology Knowledge Interchange Format, 2009

[PMZ11a] H. Pahlow, M. Müller, S.Czapla, English Grammar Online 4 U, http://www.ego4u.de/de/cram-up/grammar . Letzter Zugriff 20.01.2012

[PMZ11b] H. Pahlow, M. Müller, S.Czapla, English Grammar Online 4 U, http://www.ego4u.de/de/cram-up/grammar/adjectives-adverbs/adjective-or-adverb Zuletzt besucht am 29, letzter Zugriff am 16.12.11

[Scha10] U. Schade, Computerlinguistik Vorlesung, Fraunhofer-Institut für Kommunikation, Informationstechnik und Ergonomie, Bonn, 2010.

[Simo11] R. L. Simmons, Grammar Bytes, The Subordinate Clause, http://www.chompchomp.com/terms/subordinateclause.htm, Zuletzt zugegriffen am 25.12.2011

[SNL01] SOON W. M., NG Hwee T., LIM Daniel Chung Y.: A machine learning approach to coreference resolution of noun phrases. In: Computational Linguistics 27, 2011

[Stan08] The Stanford Natural Language Processing Group, Stanford NER CRF FAQ, http://nlp.stanford.edu/software/crf-faq.shtml#c. Zulestzt besucht am 29. Januar 2012.

[Stuf11] How Stuff Works?, www.howstuffworks.com, letzter Zugriff 20.12.11

[Sumo11] Suggested Upper Merged Ontology (SUMO), Ontologyportal, http://www.ontologyportal.org/, letzter Zugriff am 04.09.2011.

[Ulle09] C. Ullenboom, Java ist auch eine Insel, ISBN 978-3-8362-1371-4, Galileo Computing, 2009

[Verb10] University of Colorado Boulder, http://verbs.colorado.edu/verb-index/vn/murder-42.1.php, letzter Zugriff am 22. August 2011

[Voge11] Preference Pages, http://www.vogella.de/articles/EclipsePreferences/article.html, letzter Zugriff 18.11.2011

[W3C04] Wordnet in RDFS and OWL, www.w3.org/2001/sw/BestPractices/WNET/wordnet-sw-20040713.html, W3C, 2004, letzter Zugriff 13.09.2011

[Wies11] J. Wiessinger, Die Flüssigkeitskühlung, http://www.kfztech.de/kfztechnik/motor/kuehlung/wasserkuehlung.htm, letzter Zugriff am 02.10.2011

[Wiki11] Wikipedia, Die freie Enzyklopädie, http://de.wikipedia.org/wiki/Hyperonym, Zuletzt besucht am 30.12.2011.

[Word11] WordNet, A lexical Database for Englisch, Princeton University, http://wordnetweb.princeton.edu/perl/webwn, letzter Zugriff am 21. Juni 2011

A. Installation des AA und der Komponenten [Land10]

AA wird über den Eclipse Update Manager heruntergeladen. Hierfür wir die folgende URL im Help > Install New Software > Edit Site > Location eingetragen: http://svn.ipd.uni-karlsruhe.de/repos/koerner/mx/mx_update_site/site.xml. Nun kann AA über zwei Wege gestartet werden, wobei hierbei eine (z).txt Datei gebraucht wird.

- *Zum Starten des AA über Eclipse:* Zunächst wird in Eclipse die Perspektive „SALEMX“ ausgewählt. Fallst z.txt schon im Verzeichnis liegt, so kann man mit Rechstcklick > Process with AA das Programm starten.

- *Zum Starten des AA über Konsole:* Hier wird lediglich der folgende Befehl im gleichen Verzeichnis der Plain-Text Datei eingegeben:

  - java –jar AutoAnnotator-jar-with-dependencies.jar –inFile z.txt –outFile z.SALE –function annotate –config y.props.

Beim Ausführen des Programms werden bezüglich des NER-Servers mehrere Fehler angezeigt, falls man ihn vorher nicht gestartet hat. Man sollte daher den folgenden Befehl [Stan08] in einem seperaten Cmd-Fenster eingeben:

- java –mx500m –cp stanford-ner-with-classifier.jar edu.stanford.nlp.ie.NERServer –loadClassifier classifier/all.3class.distsim.crf.ser.gz –port 19201.

Doch die gerade eben verwendete Jar-Datei muss zuerst erzeugt werden. Die gewünschte Version kann man vom Stanford NLP Group herunterladen. Für den Bau des Jars wird dann folgendes eingegeben:

- cp stanford-ner.jar stanford-ner-with-classifier.jar jar –uf Stanford-ner-with-classifier.jar classifiers\all.3class.distsim.crf.ser.gz

Die Installation der anderen Komponenten (JavaRap, Charniak Parser) sowie des NER-Servers findet man im Anhang F, G und H von [Land10].

B. Subphrase-Erkennung [Simo11]

Subordinate Clause → Subordinate Conjunctions:

after	once	until
although	provided that	when
as	rather than	whenever
because	since	where
before	so that	whereas
even if	than	wherever
even though	that	whether
if	though	while
in order that	Unless	why

Subordinate Clause → Relative Pronouns:

that	who	whose
which	whoever	whosever
whichever	whom	whomever

C. Art und Position der Adverbien [PMZ11]

Adverbien der Art und Weisse:

(slowly, carefully, completely)

Adverbien stehen entweder hinter dem direkten Objekt oder falls keines existiert hinter dem Verb. In beiden Fällen also auf jeden Fall irgendwann **nach** dem Verb. Betrachtet wird Satz 11.

…	Prädikat	Adverb
The outlet to the radiator	is **blocked**	**completely.**

Adverbien des Ortes:

(here, there, behind)

Die Position dieser Adverbien sind analog zu Adverbien der Art und Weisse.

Adverbien der Zeit:

(recently, now, then, yesterday, when)

Adverb	Prädikat	Prädikat	Objekt
When	the temperature of the coolant	**goes**	above a set point

Diese Art von Adverb befindet sich entweder am Anfang oder am Ende eines Satzes. Anderes Beispiel siehe Satz 12 (… and then).

Häufigkeitsadverbien:

(always, never, seldom, usually)

Diese Adverbien stehen mit wenigen Ausnahmen hauptsächlich vor dem Vollverb. Siehe Beispielsatz 3 (…usually have engine driven cooling fans).

Gradadverbien:

(fairly, rather, very, really, quite)

Positionierung wie Häufigkeitsadverbien.

Konjunktive Adverbien:

(however, otherwise, still, therefore, whenever, furthermore, moreover, even though, also, when, anyhow, accordingly)

Diese Adverbien verbinden mehrere zusammenhängende Sätze.

Siehe Anhang B

D. Technische Sätze [Stuf11]

Nr.	Text	Kategorie
Satz 1	The fans are controlled either with a thermostatic switch or by the engine computer, and they turn on when the temperature of the coolant goes above a set point.	Fan
Satz 2	Satz 1 + They turn back off when the temperature drops below that point.	Fan
Satz 3	Rear-wheel driven cars with longitudinal engines usually have engine-driven cooling fans.	Fan
Satz 4	The cooling fan has to be controlled so that it allows the engine to maintain a constant temperature	Fan
Satz 5	When the wax melts, it expands significantly, pushing the rod out of the cylinder and opening the valve.	Thermostat
Satz 6	A rod connected to the valve presses into this wax.	Thermostat
Satz 7	The cylinder, which is located on the engine-side of the device, is filled with a wax that begins to melt at around 180 Fahrenheit.	Thermostat
Satz 8	Buy the time the coolant reaches 93 to 103 Celcius, the thermostat is open all the way.	Thermostat
Satz 9	Once the temperature of the coolant rises to between 82 and 91 Celcius, the thermostat starts to open, allowing fluid to flow through the radiator.	Thermostat
Satz 10	All of the coolant is recirculated back through the engine. →Probleme	Thermostat
Satz 11	At low temperatures, the outlet to the radiator is blocked completely.	Thermostat
Satz 12 a	The Thermostat allows the engine to heat up quickly, and then to keep the engine at a constant temperature. → Probleme	Thermostat
Satz 12 b	Satz 12 a + It does this by regulating the amount of water that goes through the radiator. → Probleme	Thermostat
Satz 13	The cooling system is composed of radiator, pressure_cap and fan.	Cooling System
Satz 20	A WHOIS server listens on TCP port 43 for requests from WHOIS clients. The WHOIS client makes a text request to the WHOIS server, then the WHOIS server replies with text content. All requests are terminated with ASCII CR and then ASCII LF.	Whois_Server [Dai97]
Satz 21	The traffic lights allow pedestrians to cross the road safely.	[Heim06]

E. Thematische Rollen [Gelh10]

ACT	Actus	Eine Handlung
AG	Agens	Ein Handelnder
BEN+/-	Beneficiens	Der Nutznießer ein Handlung
CAU+/-	Causa	Ein Grund
COM	Comes	Ein Begleiter
COMP	Comparand	Etwas, mit dem verglichen wird
COMPII	Coistmpariens	Etwas, das verglichen wird
CONS+/-	Consequentia	Das, was gilt, wenn eine Bedingung erfüllt ist
CONT	Contrarius	Ein Gegner
CONTII	Contrariens	Jemand, der einen Gegner hat
CREA	Creator	Synonym für *agens,* vor allem im Zusammenhang mit *opus*
CRIT	Criterium	Ein Vergleichskriterium
CUR	Currens	Das Gegenwärtige, Laufende
DON	Donor	Jemand, der etwas gibt
DUX	Dux	Jemand, der begleitet wird
EQD	Equal-drop	Technische Rolle
EQK	Equal-keep	Technische Rolle
EXP	Experior	Jemand, der etwas erfährt
FAU+/-	Fautor	Jemand, der einem einen Vorteil verschafft
FAV+/-	Favor	Ein Vorteil
FIC	Fictum	Eine Funktion oder Rolle, die jemand oder etwas spielt
FIN	Fingens	Jemand, der eine Rolle oder Funktion einnimmt

FREQ	Frequens	Eine zeitlich Häufigkeit
HAB	Habitum	Etwas, das jemand hat oder das ihm gehört
INST+/-	Instrumentum	Ein Hilfsmittel bei einer Tätigkeit
INT	Intentio	Eine Absicht
IUS+/-	Ius	Ein Recht (Anrecht), das jemand hat
IUSII	Iurens	Jemand, der ein Recht hat, der etwas derf
LDEST	Locus destinatio	Ein Ort, wo etwas hin geht oder bis zu dem es reicht
LDIM	Locus dimensio	Eine Strecke (Länge, Höhe, Breite,…..)
LIM	Limes	Ein Pfad, ein Weg, den etwas nimmt
LOC	Locus	Ein Ort
LORIG	Locus oirgo	Ein Ort, wo etwas herkommt oder wo es anfängt
LTRANS	Locus transitum	Etwas, das sich durch dem Raum bewegt (oder bewegt wird)
MAG	Magister	Ein Lehrer, jemand, der einen anderen in die Lage versetzt, etwas zu tun
MOD	Modus	Die Art, wie jemand etwas tut
NOT	Noito	Eine Erfahrung oder eine Empfindung, die jemand macht
OBL	Obligens	Jemand, der jemand anderen verpflichtet, etwas zu tun
OMN	Omnium	Ein/das Ganze
OPUS+/-	Opus	Ein Werk, etwas, das geschaffen (+) oder zerstört (-) wird
PAR	Pars	Ein Teil eines Ganzen
PAT	Patiens	Ein Behandelter, das Ziel einer

		Handlung
PERM	Permitens	Jemand, der etwas erlaubt
POSS	Possesor	Der Besitzer einer Sache, der Halter, der „Haber“
POT+/-	Potentia	Ein Können, eine Fähigkeit
POTII	Potens	Jemand, der etwas kann
PRAE	Praecedens	Der Vorgänger, Vorläufer, was zuvor war
PROP	Proportiens	Ein Kriterium, an dem die Größe eines Vergleichskriteriums bestimmt wird
QUAL	Qualitas	Eine Qualität, eine Beschaffenheit hat
QUALII	Aualificatus	Etwas, dass eine Qualität hat, dessen Beschaffenheit beschrieben wird
RECP	Recipient	Jemand, der etwas bekommt oder erhält
REQ+/-	Requisitum	Eine Anforderung, eine Pflicht
REQII	Requirens	Jemand, der eine Pflicht hat, an den eine Anforderung gestellt ist
STAT	Status	Ein Zustand
STIM	Stimulus	Jemand oder etwas, das eine Erfahrung oder Empfindung stimuliert
SUB	Substituens	Etwas, das ersetzt wird
SUBII	Substitutus	Etwas, das etwas anderes ersetzt
SUCC	Succedens	Der Nachfolger, etwas, das danach kommt
SUM	Sumtio	Eine Bedingung, eine Voraussetzung
TDEST	Tempus destinatio	Ein Zeitpunkt bi zu dem etwas ist, etwas war oder sein wird

TDIM	Tempus dimensio	Eine Zeitspanne, ein Zeitraum
TEMP	Tempus	Ein Zeitpunkt
THE	Thema	Ein Thema, der Inhallt
THEII	Thematus	Etwas, das ein Thema hat
TORIG	Tempus origo	Ein Zeitpunkt, seit dem etwas ist oder ab dem etwas war oder sein wird
TRANS	Transitus	Ein Zusandsübergang
TTEANS	Tempus transitum	Etwas, das sich durch die Zeit bewegt oder eine zeitliche Erstreckung hat
VOL+/-	Voluntas	Ein Wille, etwas, das jemand will
VOLII	Volens	Jemand, der etwas will

F. Das Penn-Tagset [MSM04]

POS Tag	Description	Example
CC	Coordinating conjunction	and
CD	Cardinal number	1,third
DT	Determiner	The
EX	Existential there	There is
FW	Foreign word	D'hoevre
IN	Preposition/subordinating conjunction	In,of,like
JJ	Adjective	Green
JJR	Adjective,comparative	Greener
JJS	Adjective, superlative	Greenest
LS	List marker	1)
MD	Modal	Could,will
NN	Noun,singular or mass	Table
NNS	Noun plural	Tables
NNP	Proper noun, singular	John
NNPS	Proper noun, plural	Vikings
PDT	Predeterminer	Both the boys
POS	Possessive ending	Friend's
PRP	Personal pronoun	I, he, it

PRP $	Possessive pronoun	My, his
RB	Adverb	However, usually, naturally, here, good
RBR	Adverb, comparative	Better
RBS	Adverb, superlative	Best
RP	Particle	Give up
TO	To	To go, to him
UH	Interjection	Uhhuhhuhh
VB	Verb, base form	Take
VBD	Verb, past tense	Took
VBG	Verb, gerund/present participle	taking
VBN	Verb, past participle	Taken
VBP	Verb, sing.present, non-3d	Take
VBZ	Verb, 3rd person sing. Present	Takes
WDT	Wh-determiner	Which
WP	Wh-pronoun	Who,what
WP $	Possessive wh-pronoun	Whose
WRB	Wh-abverb	Where, when

G. TypedDependencies [Land10]

dep	dependenct	**amod**	adjectival modifier
aux	auxiliary	**appos**	Appositional modifier
auxpass	Passive auxiliary	**advcl**	Adverbial clause modifier
Cop	Copula	**purpel**	Purpose clause modifier
arg	argument	**det**	Determiner
agent	agent	**predect**	Predeterminer
comp	complement	**preconj**	Preconjunct
acomp	cdjectival complement	**infmod**	Infinitival modifier
Attr	attributive	**partmod**	Participial modifier
ccomp	causal compl. With int. Subj.	**advmod**	Adverbial modifier
xcomp	clausal compl. With ext. subj.	**neg**	Nagation modifier
compl **obj**	complementizer object	**remod** **quantmod**	Relative clause modifier quantifier modifier
dobj	direct object	**tmod**	Temporal modifier
iobj	indirect object	**measure**	Measure-phrase modifier
pobj	object of preposition	**nn**	Noun compound modifier
mark	marker	**num**	Numeric modifier
rel	Relative (word introd. A rcmod)	**number**	Element of compound number

subj **nsubj**	Subject nominal subject	**prep** **poss**	Prepositional modifier possession modifier
nsubjpass	Passive nominal subject	**possessive**	Possessive modifier (‘s)
csubj	Clausal subject	**prt**	Phrasal verb particle
csubjpass	Passive clausal subject	**parataxis**	Parataxis
cc	Coordination	**punct**	Punctuation
conj	Conjunct	**ref**	Referent
expl	Expletive (expletive there)	**sdep**	Semantic dependent
mod	Modifier	**xsubj**	Controlling subject
abbrev	Abbreviation modifier		

H. Rollen von Ontologie-Ressourcen

FrameNet: [(vgl. FrameNet & Gelh10)]

Wie in Grundlagen schon erwähnt besteht jedes Frame aus FrameNet Elementen (FEs) und Lexical Units (LUs). Exemplarisch werden nur fünf FEs bzw. Rollen aufgelistet.

Thematische Rollen vom FrameNet	**Thematische Rollen für SALE**
Goal	-
Agent	AG
Manner	-
Duration [Dur]_	TDIM
Temperature_Setting	-

VerbNet: [(vgl. VerbNet & Gelh10)]

Thematische Rollen vomVerbNet	**Beschreibung von Rollen aus VerbNet**	**Thematische Rollen für SALE**
Agent	a animate subject	AG
Attribute	Attribute of patient. „**The price of oil** soared.“	-
Patient	used for participants that have been affected in some way	PAT
Instrument	used for objects that come in contact with an object and cause some change in them	INST+/-

Location, Destination, Source	used for **spatial** locations	LDIM
Location	Unspecified Place, Destination …	LOC
Destination	end point of the motion	LDEST
Source	start point oft the motion	LORIG
Material	Start point of transformation	~PAT
Product	end result of transformation	OPUS+
Time	Express the time	TDEST, TORIG, TEMP
Topic	topic of communication verbs	-
Theme		THE
Recipient	target of the transfer	RECP
Extend	The range or degree of change	-

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Unterschrift:

Stuttgart, den 01.02.2012