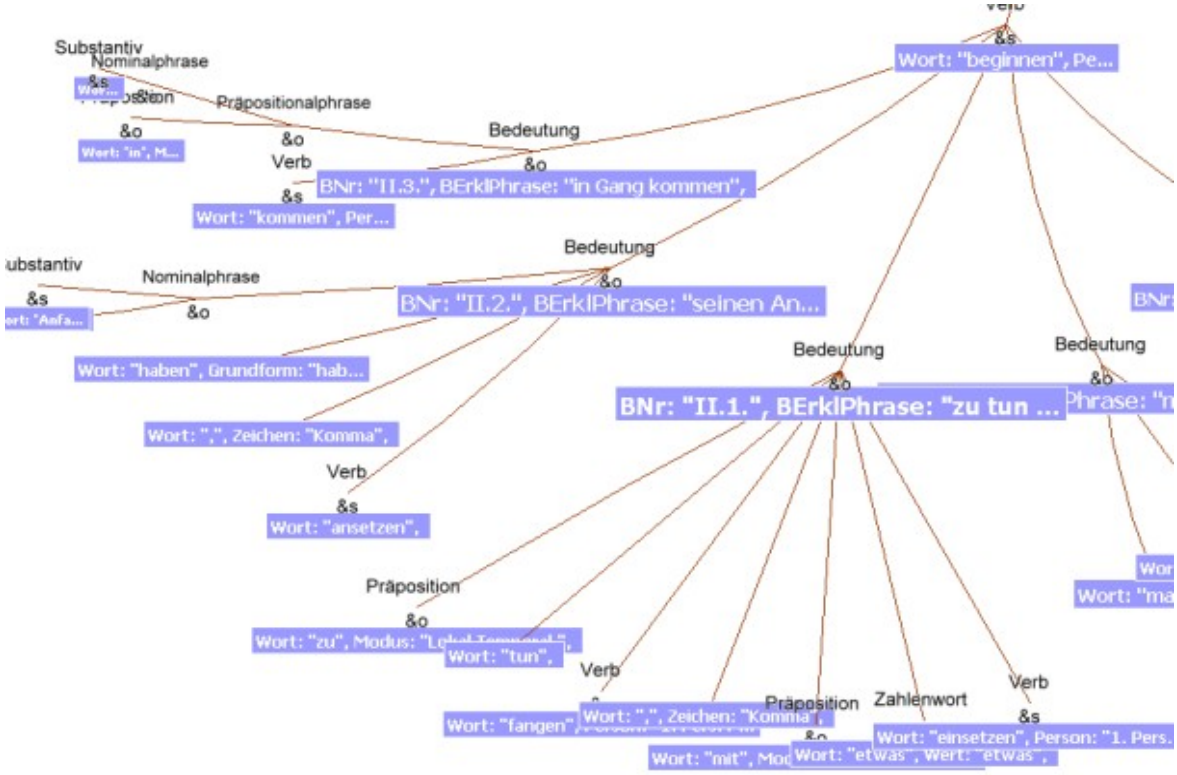
Fachhochschule Köln
University of Applied Sciences Cologne

07 Fakultät für Informations-, Medien- und Elektrotechnik, Studiengang Elektrotechnik

Institut für Nachrichtentechnik
Labor für Informatik

M a s t e r t h e s i s



Thema: **Anwendung der Konzeption semistrukturierter Daten zur semantisch orientierten Verarbeitung von Verben der deutschen Sprache**

2. revidierte veröffentlichte KOPS Ausgabe

Dipl. Inform. (FH) Andreas Frey
Matr.-Nr.: 11013600

eMail: andreas.frey@in2con.de

Referent:
Prof. Dr. phil. Gregor Büchel

Koreferent:
Prof. Dr.rer.nat. Heiko Knospe

Abgabedatum: 26.2.2007

Hiermit versichere ich, dass ich die Masterthesis selbstständig angefertigt und keine anderen als die angegebenen und bei Zitaten kenntlich gemachten Quellen und Hilfsmittel benutzt habe.

Andreas Frey

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Änderungen und Einschränkungen der KOPS-Veröffentlichung aus Gründen des Copyrights

Um die Lesbarkeit und Qualität dieser Studie zu verbessern, wurden vom Autor einige Korrekturen gegenüber der ursprünglichen Fassung vorgenommnen.

Das für die Masterthesis realisierte Softwaresystem verwendet als Teilkomponente die Softwarelösung IPEE (Internet Pattern Extrakt Engine), die von M. Sc. Andreas Frey entwickelt wurde und zu kommerziellen Zwecken eingesetzt wird. Aus diesem Grunde werden Teile der Dokumentation der Softwarelösung IPEE aus der ursprünglichen Version der Masterthesis in dieser veröffentlichten Ausgabe nicht aufgenommen. Es ist dabei darauf geachtet worden, dass das grundlegende Thema der Masterthesis in Inhalt und Sinn nachvollziehbar geblieben ist.
Im Inhaltsverzeichnis sind die entsprechenden Kapitel und Abschnitte jedoch beibehalten und
werden im Verlauf des Buches an den jeweiligen Seiten erwähnt und mit einem Copyrightvermerk gekennzeichnet. Dies betrifft folgende Kapitel und Anhänge:

Kapitel 7 : Element-Transformation
Kapitel 8 : Design und implementiertechnische Aspekte der Extraktion
Kapitel 9 : Design und implementiertechnische Aspekte der Threadlists
Anhang A: Pattern-Extraktion

Auch sind weitere Quellcodes der Softwarelösung IPEE für die Realisierung des Prototypen zur semantisch orientierten Verarbeitung von Verben der deutschen Sprache in dieser Ausgabe nicht enthalten. Die Verwendung unterliegt dem Labor für Informatik der Fachhochschule Köln, University of Applied Sciences Cologne, Fakultät 07 für Informations-, Medien- und Elektrotechnik, Studiengang Elektrotechnik

Dies betrifft die Anhänge:

Anhang E: Anmerkungen zum Programmcode
Anhang F: Schnittstellen von IPEE
Anhang G: Java Source-Code
Anhang H: Threadlist-XML-Datei
Anhang I: Patternlist-XML-Dateien

Dank

Mit der Idee, mich mit semantisch orientierter Verarbeitung zu befassen, habe ich mich an Professoren gewandt, die bereit sind, mich mit diesem Thema innerhalb meines Masterstudiengangs "Information Engineering" zu betreuen.

Ich möchte mich besonders bei Professor Dr. phil. Gregor Büchel bedanken, der sich für dieses Thema bereit gestellt hat, und mit dem ich auf eine gemeinsame Linie gekommen bin, ein Lösungskonzept für ein interessantes Experiment der semantischen Verarbeitung zu entwickeln. Es hat sehr viel Spaß gemacht, unsere Ideen zu integrieren und in eine Anwendung umzusetzen.
Ebenso möchte ich dem Koreferent Professor Dr. rer. nat. Heiko Knospe, der mir ebenfalls mit Rat und Tat zur Seite stand, für das Interesse an diesem Thema danken.

Nicht zuletzt möchte ich meinen Kunden danken, die mit ihren Anfragen wesentliche Impulse gegeben haben, insbesondere für den Auftrag der Softwarelösung IPEE (Internet Pattern Extract Engine), die ein Grundstein für diese wissenschaftliche Arbeit ist.

Ergebnisse dieser Arbeit konnten im Rahmen einer Forschungsarbeit von Professor Büchel zur maschinellen Verarbeitung der Bedeutungserklärungen von Verben angewendet werden.[1] Dies freut mich sehr, da dadurch neue Erkenntnisse gewonnen werden.

1 Büchel, Gregor: „Strukturierung von Bedeutungserklärungen mit XMLTechnologien am Beispiel von Verben“, http://www.ifk.uni-bonn.de/forschung/abteilung-sprache-und-kommunikation/ikp-arbeitsberichte-neue-folge/ikpab-nf22.pdf (2.2008)

Einleitung

Das heutige World Wide Web (WWW) hat die Möglichkeit des Zugangs zu Informationen revolutionär verändert. Die Daten des Internets sind sehr vielfältig, und die Informationsmenge steigt rapide an. Die große Menge der Daten und auch die unterschiedliche Art der Formatierung des Webs stellen den Anwender jedoch häufig vor Schwierigkeiten, relevante Informationen zu finden. [2]

Aus diesem Grunde ist der Erfinder des Internets, Tim Berners-Lee, schon fünf Jahre mit der Entwicklung des sogenannten Semantic Web beschäftigt. [3] „Als marktreif sieht er es selbst jedoch noch nicht: ‚Die meisten Unternehmen beginnen gerade erst, sich mit dem Semantic Web zu befassen. Die Softwareentwickler sind wesentlich weiter...'"[4]

Für die Entwicklung intelligenter Systeme scheint es jedoch auch einen Antitrend zu geben: Tim O'Reilly, ein Vordenker des modernen Internets, gründete die erste kommerzielle Webseite; der heute sehr stark diskutierte Begriff Web 2.0 stammt ebenfalls von ihm.[5] Er äußert sich folgendermaßen: „Jahrelang habe man versucht, ´Intelligenz in Systeme zu packen, die unabhängig von uns existieren können´. Heute gehe es aber vielmehr darum, Applikationen durch eine gemeinsame menschliche Anstrengung zu erweitern..."[6]

Aus den Aussagen dieses Experten kann zwar nicht geschlossen werden, dass das Semantik Web in einigen Jahren mit der Verbreitung wie das heutige World Wide Web existieren wird. Tim Berners-Lee erwartet das Web 3.0 jedoch in den nächsten 15 Jahren.

So ist eine mögliche Weiterentwicklung und Verbreitung nicht auszuschließen.
Um mich darauf vorzubereiten und um grundlegende Möglichkeiten und auch Einschränkungen kennen zu lernen, habe ich mich entschlossen, mich mit Teilaspekten von semantisch orientierten, verarbeitenden Systemen zu beschäftigen, die für die Realisierung von Semantic Web notwendig sind.

2 Beckstein, Sack, Peter: „Semantic Web", http://www.informatik.unijena.de/~hpeter/Lehre/Seminare/Semantic_Web_SS06/Seminarankuendigung/Seminarankuendigung.html (2.2007)
3 Vgl. Kling: „Tim Berners-Lee verkündet das Web 3.0", http://de.theinquirer.net/2006/09/24/tim_bernerslee_verkuendet_das.html (2.2007)
4 Kling: „Tim Berners-Lee verkündet das Web 3.0", http://de.theinquirer.net/2006/09/24/tim_bernerslee_verkuendet_das.html (2.2007)
5 Vgl. Hage: „Der Web-Meister" http://www.manager-magazin.de/it/artikel/0,2828,449911,00.html (2.2007)
6 Hage: „Computer werden halb menschlich", http://www.manager-magazin.de/it/artikel/0,2828,449911-4,00.html (2.2007)

Heute existieren bereits grundlegende standardisierte Technologien des W3C-Konsortiums, wie RDF[7] und OWL[8], die auf der Notation XML basieren. Die Notation XML wird für die Ablage und Verarbeitung von semistrukturierten Daten eingesetzt, was innerhalb dieser Masterthesis in Kapitel 2 „Konzeptionen von Datenstrukturen“ erklärt wird. Die Konzeption von semistrukturierten Daten findet innerhalb der semantisch orientierten Verarbeitung im hier vorgestellten Projekt ihre Anwendung.

Die grundlegende Idee ist, einen Prototyp zu entwickeln, der semantisch ähnliche Sätze in Beziehung setzt. Da der Aufwand für eine semantische Verarbeitung von natürlichen Sprachen unerschöpflich ist, wurde beschlossen, sich auf die semantisch orientierte Verarbeitung von Verben zu konzentrieren.
Die semantische Beziehung wird durch einen semantischen Graphen, der im folgenden als Verbfeld-Graph bezeichnet wird, beschrieben.

Sätze der deutschen Sprache sind semistrukturierte Daten, da sie mit Hilfe von Erkennungsmustern (Patterns) und einem verarbeitenden Verfahren in wohlgeformte semistrukturierte Daten transformiert werden können, die wiederum als OEM-Graph repräsentiert werden und in Notationen wie ssd oder XML dargestellt werden können.
Im ersten Schritt wird ein eingegebener Satz der deutschen Sprache wort- und phrasen-analysiert. Dabei findet bereits diese Transformation statt.
Den Wörtern wird eine Wortart und weitere Attribute wie Person, Numerus, Kasus, Genus etc. zugeordnet. Innerhalb der Phrasen-Analyse werden Nominal- und Präpositionalphrasen analysiert. Somit können vom Betrachter aus Musterübereinstimmungen innerhalb der Phrasen von Sätzen mit semantisch ähnlichen Verben festgestellt werden.

Für die Generierung eines Verbfeld-Graphen werden folgende Schritte durchgeführt:

1. Eingabe eines Satzes der deutschen Sprache
2. Wort- und Phrasen-Analyse (Transformation in wohlgeformte, semistrukturierte Daten)
3. Recherchieren von Bedeutungserklärungen des Verbs aus dem Satz
4. Wort- und Phrasen-Analyse der Bedeutungserklärungen
5. Anhängen der Bedeutungserklärungen an das Verb innerhalb des Verbfeld-Graphen

Die Bedeutungserklärungen enthalten ebenfalls Verben, zu denen wiederum Bedeutungserklärungen angefügt werden können. Somit können die Schritte 3 bis 5 rekursiv durchgeführt werden. Die Anzahl der Rekursionsebenen ist innerhalb dieser experimentellen Auswertung auf zwei Ebenen beschränkt.

Nach Durchführung dieses Verfahrens können weitere Sätze eingegeben werden und der Verbfeld-Graph erweitert sich.

7 Herman, Swick, Brickly: „Resource Description Framework (RDF)“, http://www.w3.org/RDF/ (2.2007)
8 L. McGuinness, Harmelen: „OWL Web Ontology Language“, http://www.w3.org/TR/owl-features/ (2.2007)

Sollte nun im zweiten Satz ein Verb verarbeitet werden, das im entstehenden Graphen schon existiert, wird dieser nicht ein weiteres Mal mit Bedeutungen versehen, sondern es wird eine Referenz zwischen dem gerade verarbeiteten Verb und den schon vorhandenen gezogen.

Das Verb innerhalb einer Bedeutungserklärung wird als Oberbegriff (Hyperonym) betrachtet. Eine semantische Beziehung soll damit zum Ausdruck gebracht werden, dass Verben zweier oder mehrerer Sätze einem Oberbegriff zugeordnet werden.

Diese Aufgabe und auch die Internet-Zugriffe für die Erkennung bestimmter Wortarten und das Recherchieren von Bedeutungen wird durch meine entwickelte Softwarelösung IPEE (Internet Pattern Extract Engine) durchgeführt. Ursprünglich wurde diese Software als kundenspezifischer Auftrag für die Extraktion von Fahrzeugdaten für Automobil-Handelsbörsen erarbeitet. Während der Entwicklung habe ich Wert darauf gelegt, dass diese Engine auch als allgemein einsetzbare Meta-Suchmaschine und Extraktionstool genutzt werden kann, die mit Hilfe von Patternlist- und Threadlist-XML-Datei anpassbar (customierbar) ist.
IPEE erkennt mit Hilfe von Patterns (Erkennungsmuster) Wörter, Endungen, Nominal- und Präpositionalphrasen und extrahiert die relevanten Inhalte (Wortarten, Deklinationen und Lesarten, die die Bedeutungserklärungen enthalten) aus dem Internet.
Übergeordnet wird diese Software anhand von Threadlists gesteuert, die Abläufe für den Internetzugriff und die Extraktionsverfahren beschreiben.

Innerhalb des praktischen Aufgabenteils wird für die Darstellung des Verbfeld-Graphen das Open Source Tool Treebolic[9] eingesetzt, mit dem es - mit ein paar Einschränkungen - möglich ist, die wort- und phrasen-analysierten Sätze und den Verbfeld-Graph zu visualisieren.

Nach Durchführung des beschriebenen Vorgangs für die Generierung des Verbfeld-Graphen wird dieser als XML-Datei exportiert (in ein Verzeichnis geschrieben), die einer Dokument Type Definition (DTD) für den Treebolic-Browser unterworfen ist.

Als zentrales User-Interface wurde zusätzlich von mir die Oberfläche SentenceView entwickelt. Diese ist für den Anwender ähnlich wie ein Browser konstruiert.
Entwickelt wird die Programmiersprache Java mit der J2SE1.5.x[10]; (auf Verwendung weiterer APIs[11] wird im folgenden durch Literaturverweise hingewiesen).

Vom Anwender aus kann nach Eingabe von Sätzen oder Wörtern aus der deutschen Sprache innerhalb von SentenceView die Wort- und Phrasen-Analyse und die Verbfeld-Graph-Generierung ausgeführt werden.

9 Treebolic, http://sourceforge.net/projects/treebolic/ (2.2007)
10 Sun Microsystems: „JavaTM 2 Platform Standard Edition 5.0API Specification“, http://java.sun.com/j2se/1.5.0/docs/api/ (2.2007)
11 Application Programming Interface - Schnittstelle für ein Softwaresystem an Funktionen des Betriebssystems oder weiteren Systemen

Das Ergebnis der Wort- und Phrasen-Analyse kann entsprechend dieser beiden Stufen anhand von graphischen Elementen für Tabellen-Ansichten betrachtet werden. Zusätzlich werden die Lesarten, die u. a. die Bedeutungserklärungen des Verbs des eingegebenen Satzes enthalten, angezeigt.

Während der Arbeit mit SentenceView wird automatisch ein zusätzliches Bedeutungs-Lexikon für Verben gepflegt. Das Lexikon wird bei der Verarbeitung der eingegebenen Sätze und bei der Recherche der Bedeutungen automatisch um noch nicht eingetragene Verben erweitert. Dieses Lexikon wird auch verwendet, um die Generierung von Verbfeld-Graphen zu beschleunigen. Für die Recherche von Verben werden somit Zugriffe auf das Internet und die Wort- und Phrasen-Analyse eingespart.

Im folgenden soll eine kurze Kapitelbeschreibung gegeben werden:

- In Kapitel 1 **„Was bedeutet Semantik?“** wird der Begriff „Semantik“ für diese Arbeit definiert - und erklärt, wie semantisch orientierte, verarbeitende Prozesse auf abstrakter Ebene gestaltet sind.

- Die Struktur der Daten, die für die semantisch orientierte Verarbeitung zum Einsatz kommen, basieren auf der Konzeption der semistrukturierten Daten. Diese wird im Kapitel 2 **„Konzeptionen von Datenstrukturen“** angesprochen. Grundlegend existieren die drei wesentlichen Formen von Datenstrukturen: strukturiert, semistrukturiert und nicht strukturiert, die in diesem Kapitel definiert werden und deren Anwendung beschrieben wird.

- Darauf folgen in den Kapiteln 3 und 4 **„Wort-Analyse“** und **„Phrasen-Analyse“** die grundlegenden Aspekte für diese genannten Analyseverfahren.

- In Kapitel 5 **„Anwendung der Konzeption semistrukturierter Daten für Satzmuster“** wird darauf eingegangen, wie semistrukturierte Daten innerhalb der Wort- und Phrasen-Analyse angewandt werden und welche Stadien diese durchlaufen.

- In Kapitel 6 **„Realisierung“** wird die Softwarelösung IPEE beschrieben und Argumente für deren Einsatz bei der Wort- und Phrasen-Analyse aufgezeigt. Darauf erfolgt die Beschreibung der Oberfläche SentenceView mit ihren Anwendungsfällen und Beschreibung weiterer Klassen für die Verbfeld-Graph-Generierung, die Verwaltung des Bedeutungslexikons für Verben und Durchführung von Aufrufen auf Klassen von IPEE.

- Auf die Software IPEE - zumindest auf die Komponenten, die innerhalb dieser Masterthesis eingesetzt werden - wird in den folgenden Kapiteln genauer eingegangen.
  Die Softwarelösung IPEE arbeitet mit XML-Dokumenten für die Beschreibung von Patterns und übergeordneten Abläufen und verarbeitet HTML-Dokumente. Die Elemente innerhalb dieser Dokumente werden in Instanzen umgewandelt. Die

Beschreibung dieses Prozesses erfolgt in Kapitel 7 **„Element-Transformation“.**

- Das Kapitel 8 **„Design und implementiertechnische Aspekte der Threadlists“** beschreibt die Konzeption, die zur Abarbeitung einer übergeordneten, kontrollstrukturorientierten Anweisungsliste dient.

- Das algorithmische Verfahren der verschiedenen Pattern-Extraktions-Komponenten und Beschreibung der Klassenstruktur ist im Kapitel 9 **„Design und implementiertechnische Aspekte der Extraktion“** wiederzufinden.

- Im Kapitel 10 **„Implementiertechnische Aspekte der Wort- und Phrasen-Analyse“** erfolgt die Beschreibung der Patterns für die Wort- und Phrasen-Analyse. Eine vollständige Beschreibung der Patterns und Threadlists würde den Rahmen dieser Masterthesis sprengen; daher soll nur auf einige wenige, jedoch komplexe Patterns eingegangen werden.

- Das Kapitel 11 **„Semantisch orientierte Verarbeitung“** beschreibt Verfahren zur semantischen Verarbeitung von Verben und Satzmustern, die Realisierung eines Verfahrens und wertet experimentelle Ergebnisse aus.

- In Kapitel 12 **„Hyperbolische Bäume“** wird die Darstellung des Verbfeld-Graphen anhand der hyperbolischen Bäume mit Hilfe der Open Source Software Treebolic Browser beschrieben.

- Was für Erkenntnisse, Erfahrungen und Lernergebnisse sind entstanden? Darauf soll im abschließenden Kapitel **„Resümee“** eingegangen werden. Es sollen auch Erfahrungen des Einsatzes der Softwarelösung IPEE zur Sprache kommen. Auch die Möglichkeiten und Einschränkungen von semantisch orientierten, verarbeitenden Systemen und eine Auswahl von Aspekten zur Weiterentwicklung werden dargestellt.

Vereinbarungen von Notationen

- Alle aus Programmcodes oder XML-Dateien herausgenommenen Programmzeilen, einzelne Namen von Klassen oder Instanzen oder XML-Tags bzw. Attribute-Werte-Paare von Tags werden in der Schriftart `Courier New` dargestellt.

- Ebenfalls werden auch Verzeichnispfade in `Courier New` dargestellt.

- Alle hier beschriebenen, verwendeten Dateien gehen vom Würzelverzeichnis `VerbSemantik/` aus.

- Die Zusammensetzung von Wörtern innerhalb Instanzen- und Klassennamen erfolgen nach der Java Konvention: Wörter innerhalb Klassennamen beginnen mit einem Großbuchstaben und werden zusammen geschrieben (ohne Trennzeichen). Bei Methodennamen und Namen von Instanzen wird das erste Wort klein geschrieben. Die Benennung von Objekten, Instanzen und Variablen erfolgt in englischer Sprache.

- Innerhalb der XML-Pattenlist (siehe Kapitel 9 „Design und implementiertechnische Aspekte der Extraktion“ und Band 2, Anhang A „Pattern-Extraktion“) und Threadlist-XML-Datei (siehe Kapitel 8 „ Design und implementiertechnische Aspekte der Threadlists“) wird für die Definition von Tags (Elemente), Attributen und Variablen ebenfalls die englische Sprache angewandt. Ausgenommen sind Begriffe aus der Grammatik der deutschen Sprache, die entsprechend in Deutsch deklariert sind.

  Alle beschriebenen Tags `<Tag>` aus XML-Dokumenten werden wie folgt notiert: `TagElement.`

- Wörter und Sätze der deutschen Sprache aus Beispielen bzw. Beispielsätzen und Zitaten aus Lexika und Wörterbüchern werden *kursiv* wiedergegeben - ausgenommen die Zitate im folgenden Kapitel, da dies allgemeine Zitate sind.

1 Was bedeutet Semantik?

Unter Semantik (Bedeutungslehre) versteht man im allgemeinen die linguistische Lehre von Bedeutung und Sinn sprachlicher Ausdrücke. Schon Platon begreift das sprachliche Zeichensystem als ein „tradiertes Werkzeug, mit dem wir Gegenstände, Sachverhalte und Ereignisse durch gemeinsame und insofern ‚richtige' Benennung unterscheiden".[12]
Tatsächlich spielen bereits in Platons *Kratylos* nicht nur die gemeinsame Benennung eine wesentliche Rolle, sondern vor allem auch die „konfigurativen Zusammensetzungen"[13], das heißt die Beziehungen der Wörter und Gedanken untereinander.

In gleicher Weise wie sich sprachliche Zeichen nicht nur auf singuläre Vorkommnisse, sondern vor allem auf gedankliche Verknüpfungen beziehen, läßt sich aus Platons Bestimmung der konfigurativen Zusammensetzung die Einsicht ableiten, dass nicht den Wörtern selbst Bedeutung zukommt, sondern der Sinn einer Information primär über Kontexte und vorhandenes Wissen generiert wird.
Entsprechend wird auch in der folgenden Arbeit davon ausgegangen, dass Sinngebung einer Information über ein Inbeziehungsetzen von Zeichen zu vorhandenem Wissen und Denkinhalten erfolgt.
Dabei stellt sich allerdings die Frage, wie denn Zeichen (bzw. Wörter oder Sätze der natürlichen Sprache) mit vorhandenem Wissen überhaupt in Beziehung gesetzt werden.
Der Philosoph John Locke (1632-1702), einer der wichtigsten Vertreter des englischen Empirismus, hilft hier weiter. Für seine Thesen definierte er den Begriff „Idee" folgendermaßen: „So habe ich es [das Wort „Idee"] gebraucht, um alles auszudrücken, was man unter Einbildung, Begriff, Spezies oder was sonst den menschlichen Geist beschäftigen kann, versteht." [14]
Locke unterscheidet dabei zwischen einfachen Ideen (Eindrücken, die der menschliche Geist direkt wahrnimmt) und komplexen Ideen, die sich ihrerseits aus verschiedenen Ideen zusammensetzen. Dies geschieht durch die geistigen Vorgänge *Vergleichen*, *Verbinden*, *Trennen*, *Abstrahieren*.[15]

Orientiert an den Einsichten der Semantik soll in dieser Masterthesis ein Verfahren vorgestellt werden, das experimentell einen semantischen Verarbeitungsprozess durchführt, der sich grundlegend aus den vier erwähnten Vorgängen zusammensetzt.
Eine semantische Untersuchung in dem hier verstandenen Sinne versucht also nachzuvollziehen, wie Informationen mit dem vorhandenen Wissen in Beziehung gesetzt werden. Wenn es sich zeigt, dass das gegebene Zeichen (Information) sich nicht direkt mit dem vorhandenen Wissen deckt, werden ein oder mehrere Elemente des vorhandenen Wissens und/oder die eingehenden Zeichen abstrahiert. Aus der Abstraktion einsteht eine neue Idee (neues Wissen), die das herangezogene vorhandene Wissen mit dem eingehenden Symbol verbindet. Alles weitere Wissen bleibt davon – von dem abstrahierten und dem neuen Wissen – getrennt.

12 Sandkühler: „Enzyklopädie Philosophie", Bd. 2, S. 1453
13 ebd.
14 Locke: „Versuch über den menschlichen Verstand", Bd. 1, S. 178-179
15 Vgl. ebd.

2 Konzeptionen von Datenstrukturen

Eine Konzeption, die im folgenden am intensivsten behandelt wird, ist die der semistrukturierten Daten. Diese weisen eine besondere Flexibilität gegenüber strukturierten Daten auf, da sie nicht einer Typbeschreibung (Struktur) unterliegen.
Dieser Aspekt ist in Einsatzgebieten der künstlichen Intelligenz von Relevanz, da dies ermöglicht, Wissen in Datensammlungen abzulegen, ohne dass diese einem Strukturmodell angepasst sind. Strukturmodelle können jedoch nachfolgend impliziert werden.

Es wird in diesem Kapitel zwischen den drei wesentlichen Konzeptionen - strukturierte Daten, semistrukturierte Daten und nicht strukturierte Daten - unterschieden.

Ulrike Greiner (Universität Leipzig) benennt die wesentlichen Unterschiede folgendermaßen: „Semistrukturierte Daten sind nicht streng typisiert, haben aber eine gewisse Struktur, auch wenn diese nicht sofort erkennbar ist. Daten aus relationalen oder objektorientierten Datenbanken sind streng typisiert, da sie in das Schema aus Tabellen bzw. Klassen passen müssen. Bilder oder Klänge haben überhaupt keine Struktur. Alles, was zwischen diesen beiden Extremen liegt, sind semi-strukturierte Daten.“ [16]

Bevor diese drei grundlegenden Konzeptionen genauer analysiert werden, soll der Begriff Struktur definiert werden:
Strukturen von Daten beschreiben Typen; diese enthalten den Namen und eine Menge von Attributen, die Referenzen zu weiteren Typen ausdrücken.

Strukturierte Daten besitzen somit eine strenge Typisierung; d. h. sie sind durch die Struktur von Daten festgelegt. Dies geschieht z. B. durch ein Datenbank-Schema. In diesem sind Aufbau und Datentyp für alle in der Datenbank (DB) gespeicherten Daten definiert.

Semistrukturierte Daten können ohne Typisierung existieren. Möglich ist auch, dass die Typisierung teilweise (partiell) oder vollständig vorhanden ist.

Nicht strukturierte Daten können keiner Struktur unterworfen werden. Unstrukturierte Daten sind Bitfolgen, die dem Protokoll einer Software unterliegen oder einem Dateiformat, die zwar zur Zweckbeschreibung der Daten dienen, jedoch nicht zur Typisierung der Daten.

Im folgenden sollen die drei grundlegenden Konzeptionen von Datenstrukturen präziser definiert werden.

16 Greiner, Kapitel 1.1: „Semistrukurierte Daten“, http://dbs.uni-leipzig.de/html/seminararbeiten/semSS99/arbeit3/kapitel1.html (2.2007)

2.1 Strukturierte Daten

Wie schon erwähnt, basiert eine strukturierte Datenhaltung auf ein streng typisiertes Datenmodell, das vor dem Befüllen einer Datenbank fest definiert sein muss. Dieser Aspekt findet auch in der relationalen Datenhaltung Anwendung. Die Spezifikation der Datenbank-Modelle soll in den folgenden Unterkapiteln dargestellt werden.
Die Beschreibungen sind auf ein abstraktes Niveau beschränkt. Für detailliertes Wissen sei auf spezielle Fachliteratur der sehr weit verbreiteten Anwendungsbereiche (z. B. die Ausgabe „Relationale Datenbanken“ von Hermann Sauer, Verlag Addison-Wesley) verwiesen.

2.1.1 Relationale Datenbanksysteme

Bei relationalen Datenbanksystemen wird die Struktur der Daten – d. h. Datentypen selber und die Beziehungen untereinander - anhand eines Entity-Relationship-Modells (ER-Modell) beschrieben.

Innerhalb des ER-Modells sind Typen fest spezifiziert, d. h. jedes Datenelement (Entity = deutsch Eintrag) hat eine Menge von Paaren (Tupels), die aus einem Attribut und einem Wert bestehen.

Die Attribute sind durch ihre Attributnamen fest spezifiziert. Die Werte müssen ebenfalls einem bestimmten Datentyp zugeordnet sein, wie String, Integer, Float, Char, Date, Time etc..

Zusätzlich sind auch die Referenzen (Relationen, deutsch: Beziehungen) zu anderen Typen im ER-Modell fest spezifiziert. Diese werden mit sog. Constraints, primären und sekundären Schlüsseln, hergestellt, auf deren detaillierte Spezifikation hier nicht genauer eingegangen werden soll.

Es wird somit festgelegt, welcher Datentyp zu welchem eine Beziehung haben kann. Somit ist es für eine Instanz bzw. einen Eintrag im Datenmodell nicht frei wählbar, zu welcher anderen Instanz eine Beziehung hergestellt werden kann.

Die Kardinalität legt die Referenzen zusätzlich quantitativ fest. Z.B. bedeutet die Kardinalität von 1 zu n: eine Instanz von Typ A kann mehrere Instanzen von Typ B referenzieren.[17] Weitere mögliche Kardinalitäten sind: 1 zu 1, n zu m, 1 zu 0..n, 1 zu 0..1 etc..

17 Sauer: „Relationale Datenbanken“, S. 19-24

2.1.2 Abfragen von strukturierten Daten

Ein Datenbanksystem (DBS) besteht aus einer Datenbank und einem Datenbank-Management-System (DBMS), das dafür verantwortlich ist, die Daten der Datenbank zu verwalten.

Für das Abfragen und Anlegen der Daten hat sich für relationale Datenbanksysteme der Standard Structured Query Language (SQL) durchgesetzt. In Anlehnung an SQL wurde für objektorientierte Datenbanken OQL entwickelt, dessen Notation der SQL sehr stark ähnelt.

Die SQL/OQL-Notationen erlauben es, Abfragen anhand einer „select-from-where“-Klausel zu beschreiben. Wobei das „select“, angibt, was gesucht werden soll, das „from“, von woher es kommen soll und „where“ die logische Bedingung der Suche angibt.

Für eine Suchanfrage, z. B. nach Fahrzeugmodellen in roter Farbe vom Hersteller Volkswagen, ist folgendes Select-Statement möglich:

```
select F.Modell
from Fahrzeug as F
where F.Hersteller = „Volkswagen“ and
     F.Farbe =“rot“
```

Bei der Abfrage von strukturierten Daten muss der Anwender lediglich entscheiden, welcher Datentyp (wie Kunde, Fahrzeug, Adresse) von Interesse ist. Aus der Spezifikation des ER-Modells oder Klassenmodells ist ersichtlich, wie die Eigenschaften dieser definiert sind. Dementsprechend erstellt er eine Abfrage auf den erwünschten Datentyp und definiert Bedingungen, die für die Eigenschaften der Suchmenge gelten sollen. Es müssen Attributnamen und Namen der Datentypen innerhalb der Abfrage mit der Definition des Datenmodells syntaktisch übereinstimmen.

2.2 Semistrukturierte Daten

Während bei der strukturierten Datenhaltung ein Modell zugrunde liegen muss, das Aussehen der Datenelemente (Objekte) enthält, fehlt ein solches bei semistrukturierten Daten.

Professor Dr. Georg Lausen (Dozent der Uni Freiburg) beschreibt ein Merkmal von semi-strukturierten Daten folgendermaßen: „Relationale Datenbanken verlangen die Zerlegung von zusammengehörenden Daten in unterschiedliche Relationen – semi-strukturierte Daten erlauben eine Repräsentation der Daten in ihrer eigentlichen Struktur.“[18] Das bedeutet, dass semistrukturierte Daten keinem Typenmodell unterworfen werden müssen; somit kann sich eine Datensammlung aus semistrukturierten Daten beliebig erweitern. Ein Strukturmodell

18 Lausen: „Kapitel IV, Semi-strukturierte Daten“ S. 1

kann nachfolgend impliziert werden (siehe Kapitel 2.4.4 „Ablegen von semistrukturierten Daten“).

Semistrukturierte Daten können mit Hilfe von Grammatik und Lexik in eine Form gebracht werden, die folgende Charakteristika aufweist, die auch im Rahmen des OEM-Modells (im folgenden Unterkapitel beschrieben) wiederzufinden ist:

(E1) Die Datensammlung besteht aus einer oder mehreren Folgen von Objekten.

(E2) Objekte können entweder in Attribute zerlegt werden (komplexe Objekte), oder sie sind atomare Objekte.

(E3) Atomare Objekte enthalten Werte eines bekannten, elementaren Datentyps.

Daten, die diese Eigenschaften aufweisen, können auch als wohlgeformte XML-Dateien (siehe Kapitel 2.4.2 „Die semistrukturierte Notation XML“) beschrieben werden. Auch sollen semistrukturierte Daten mit den Eigenschaften (E1), (E2) und (E3) als wohlgeformte, semistrukturierte Daten bezeichnet werden.

2.2.1 Das OEM-Modell

Um den Begriff „semistrukturierte Daten“ zu präzisieren, wird das OEM (Object Exchange Model) betrachtet.

Das OEM-Modell hat sich als de facto Standard für die Ablage von semistrukturierten Daten durchgesetzt[19] „Ursprünglich wurde es für das Tsimmis (The Stanford-IBM Manager of Multiple Information Sources) Projekt entwickelt.“

Das OEM-Modell basiert auf der Konzeption eines gerichteten Graphen. „Ein (gerichteter) Graph ist ein Paar $G=(V, E)$, hierbei ist V eine endliche Menge von Knoten und $E \subseteq V \times V$ eine Relation auf V, die Menge der Kanten.“[20] Jedoch existieren innerhalb eines OEM-Modells auch Blätter V_a, von denen aus keine Kanten wegführen können.

Die Knoten eines OEM-Graphen repräsentieren die eigentlichen Objekte, die eine eindeutige ID haben. Innerhalb eines OEM-Modells wird zwischen komplexen Objekten V_c und atomaren Objekten V_a unterschieden. Die Attribute von komplexen Objekten werden ausschließlich durch die Kanten beschrieben, die wiederum auf komplexe Objekte oder atomare Objekte referenzieren. Zusätzlich enthält der Graph eine oder mehrere Wurzeln r_1 bis r_k, von denen aus alle Objekte erreichbar sein müssen. Die atomaren Objekte repräsentieren die Blätter des Graphen, d.h. es kann nur eine Kante zu ihnen hinführen, jedoch keine wegführen. Die atomaren Objekte enthalten die Werte der

19 Vgl. Abiteboul, Buneman, Duciu: „Data on the Web“, S. 19

20 Lang: „Knoten und Kanten“, http://www.inf.fh-flensburg.de/lang/algorithmen/grundlagen/graph.htm (2.2007)

komplexen Objekte m, denen sie zugeordnet sind vom Typ Integer, String, Bild oder ähnliches.[21]

„Formal lassen sich semistrukturierte Daten also als $G=(V, E, r_1, \ldots, r_k, v)$ definieren mit:
$V=V_c \cup V_a$ die Menge der Knoten, wobei V_c die Menge der komplexen Objekte und V_a die Menge der atomaren Objekte ist;
$E \subseteq V_c \times A \times V$ die Menge der Kanten, wobei A die Menge aller Attribute ist;
r_i die Menge der Wurzeln;
$v: V_a \rightarrow D$ eine Abbildung, die atomaren Objekten Werte aus D, der Menge aller atomaren Werte, zuweist.“[22]

21 Vgl. Greiner, Kapitel 2.1.1: „Datenmodelle und Schemata für semistrukturierte Daten“, http://dbs.uni-leipzig.de/html/seminararbeiten/semSS99/arbeit3/kapitel2.html (2.2007)
22 ebd.

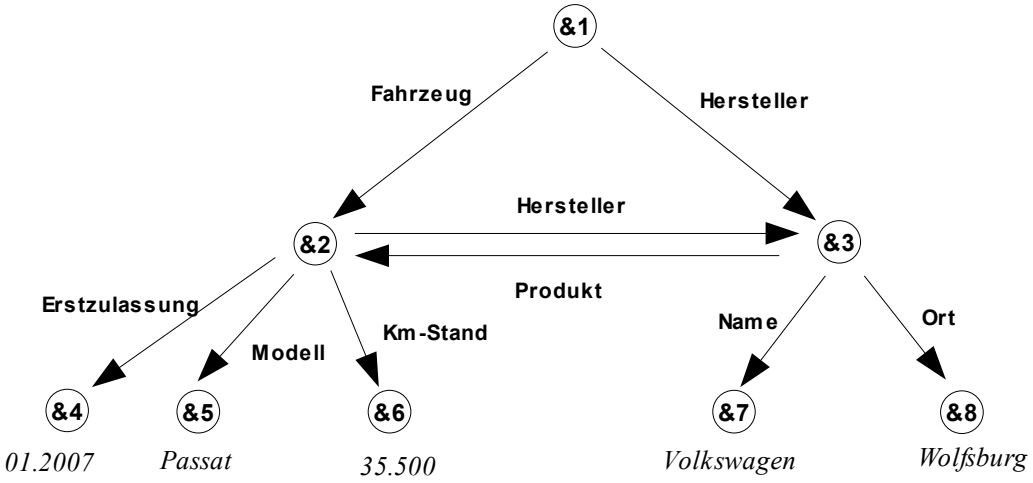


Abbildung 1: Beispiel eines OEM-Graphen

Grundsätzlich werden mit dem kartesischen Produkt $V_c \times V$ alle möglichen Kanten des Graphen definiert. Das sagt aus, dass alle komplexen Objekte mit allen Objekten referenziert sein können. Jedoch können atomare Objekte keine Referenz zu komplexen Objekten haben.

Greiner hat dieses kartesische Produkt um den Faktor A (die Menge aller Attribute) erweitert: $V_c \times A \times V$. Das drückt aus, dass die Teilmenge der Kanten jedem beliebigen Attribut zugeordnet sein kann. Somit ist den Elementen der Teilmenge der Kanten E auch ein Attribut zugeordnet; die Menge der Attribute wird nicht explizit in die Grundmenge G des Graphen aufgenommen.

Ebenso verhält es sich mit der Menge aller möglichen Werte D, die mit der Abbildung $V_a \rightarrow D$ den atomaren Objekten zugeordnet werden.

Die Wurzeln des Graphen gehören prinzipiell ebenfalls zu der Menge der komplexen Objekte. Die Wurzeln müssen die Bedingung erfüllen, dass von ihnen aus alle Knoten referenziert werden können. Um ein Beispiel eines Graphen mit zwei Wurzeln zu zeigen, kann die Wurzel in Abbildung1 weggelassen werden:

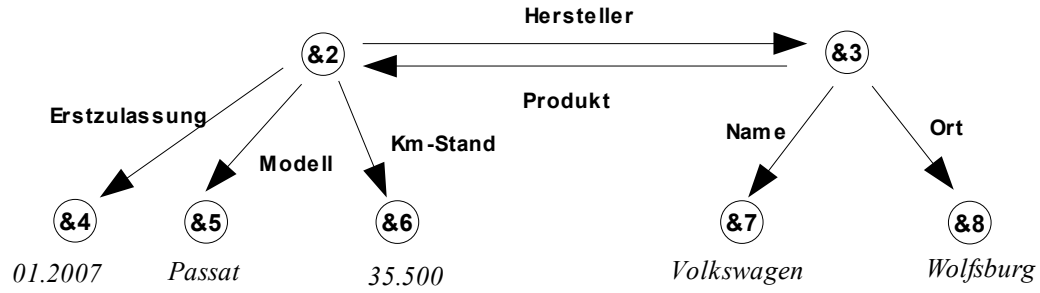


Abbildung 2: Beispiel eines OEM-Graphen mit zwei Wurzeln

Somit sind &2 und &3 Wurzeln.

2.2.2 Ist semistrukturiert nicht auch strukturiert?

Die Aussage von Professor Dr. Georg Lausen (siehe Kapitel 2.2 „Semistrukturierte Daten“) impliziert auch, dass sich semistrukturierte Daten nicht in einem strukturierten Datenbank-Modell unterbringen lassen.

Jedoch existieren Verfahren, mit denen Datentypen von semistrukturierten Daten erkannt werden können. Ein Verfahren wird im Kapitel 2.4.4 „Ablegen von semistrukturierten Daten“ beschrieben.

Wenn die Datentypen (Klassen) und damit auch die Relationen bekannt sind, hat man ein ER-Modell. Jedoch gilt für dieses Modell, dass es danach nur noch mit Daten in dieser Struktur gefüllt werden kann, nicht mehr mit weiteren semistrukturierten Daten.

Bei semistrukturierten Dateien, die in einem OEM-Modell geformt sind, kann auch behauptet werden: Die formale Beschreibung eines OEM-Modells (siehe voriges Kapitel) ermöglicht es, ein übereinstimmendes, strukturiertes Datenmodell zu erstellen, das folgendermaßen aussehen kann:

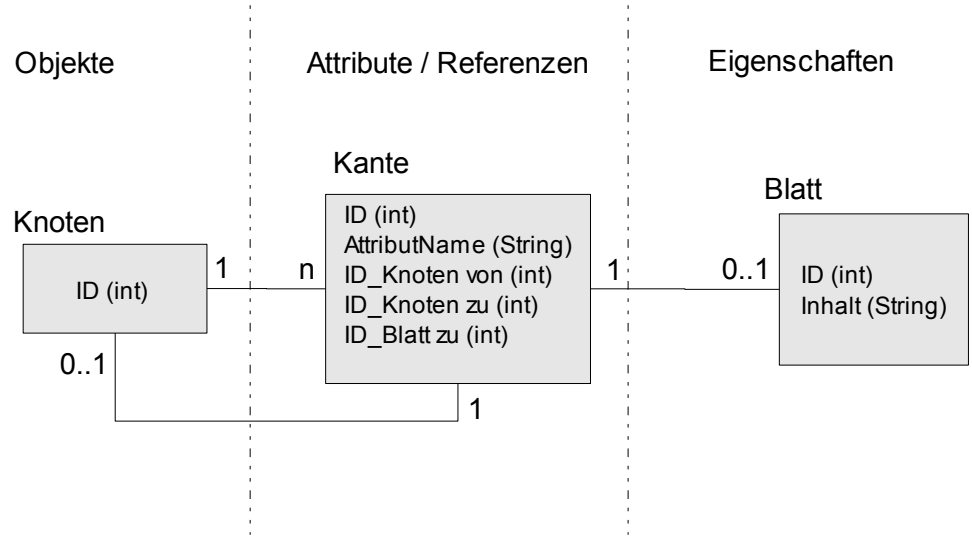


Abbildung 3: Relationales Datenmodell zur Abbildung von semistrukturierten Objekten

Dieses Datenmodell enthält nur drei grundlegende Typen: Die Knoten, die die Objekte repräsentieren, die Kanten, die Attribute bzw. Referenzen referenzieren, und Blätter, die die Eigenschaften der Referenz repräsentieren.

Somit lassen sich alle semistrukturierten Objekte eines OEM-Modells auch in dieses Datenmodell hineinschreiben. Im folgenden soll dieses OEM-DB-Modell genannt werden.

Die Aussage von Professor Dr. Georg Lausen kann mit folgender Einschränkung aufrecht erhalten werden: Semistrukturierte Daten lassen sich in kein DB-Modell hineinschreiben, außer in Modelle, die nur einen abstrakten Datentyp für alle Objekte bereit halten.

2.3 Nicht strukturierte Daten

Wie schon erwähnt, sind nicht strukturierte Daten Daten, die keine Struktur aufweisen. Greiner schreibt an dieser Stelle, dass Bilder oder Klänge keine Struktur haben. Diese lassen sich jedoch mit Anwendung bestimmter Software in Dateien ablegen, denen ein Format zugrunde liegt. Für Klangdateien sind dies z. B. .wav oder .mp3 Dateien. Diese Dateiformate unterliegen bestimmten Protokollen, die jedoch keine Strukturvorgaben enthalten. Auch ist es nicht möglich, aus diesen Daten Strukturen zu implizieren.

Des weiteren stellt sich die Frage, ob dokumentierte Sätze der natürlichen Sprache, die ein wesentlicher Bestandteil dieser Masterthesis sind, unter bestimmten Annahmen als semistrukturierte Daten betrachtet werden können.

Ohne eine Grammatik und Lexik von Wörtern, Satzphrasen und Sätzen zu Grunde zu legen,

würden in einer Natursprache geschriebene Dokumente eine unstrukturierte Folge von Zeichen darstellen.

2.4 Beschreibungsmethoden und Datenablagekonzepte für semistrukturierte Daten

Schwerpunktmäßig sollen in dieser Masterthesis die semistrukturierten Daten behandelt werden. In den folgenden Unterkapiteln sollen Notationen beschrieben werden, die semistrukturierte Daten in wohlgeformter Weise beschreiben. Dazu gehören die Notationen ssd und XML. Auch soll anhand eines Anwendungsbeispiels beschrieben werden, wie semistrukturierte Daten in eine Datensammlung abgelegt und abgefragt werden, und wie die darauf fehlenden Datentypen impliziert werden können. Anschließend werden anhand dieses Anwendungsbeispiels Möglichkeiten und Einschränkungen semantisch orientierter Abfragen auf Basis der semistrukturierten Daten mit deren existierenden Abfragemöglichkeiten beschrieben.

2.4.1 Die semistrukturierte Notation ssd

Serge Abiteboul, Peter Buneman und Dan Suciu verwenden in ihrer Ausgabe „Data on the Web“ die sog. ssd (semi-structured-data)-Notation[23], die allerdings weniger bekannt ist, als die im folgenden Unterkapitel beschriebene Notation XML. Jedoch bietet diese Notation für semistrukturierte Daten eine sehr kurze und übersichtliche Darstellung; deswegen verwende ich sie für die Ausgabe der analysierten Satzmuster (siehe Kapitel 2 „Wort-Analyse“, Kapitel 4 „Phrasen-Analyse“ und Kapitel 10 „Implementiertechnische Aspekte der Wort- und Phrasen-Analyse“).

Datensätze mit Attribut-Werte-Tupels werden folgendermaßen notiert:

```
{Hersteller: "Volkswagen" Modell: "Passat" km-Stand: "35.600" }
```

Die Werte der Attribute können nun wiederum anhand eines Unterdatensatzes definiert sein.

```
{Fahrzeug: { Hersteller: { Name : „Volkswagen" Ort: „Wolfsburg" }
Modell: "Passat" km-Stand: "35.600" } }
```

Bis jetzt ist es möglich, dass ein Element Daten bzw. Attribute-Werte-Paare enthalten kann, und diesem weitere Elemente untergeordnet sein können. Somit ermöglicht die bis jetzt vorgestellte Notation die Darstellung von Daten in Bäumen. Nach der Beschreibung der semistrukturierten Daten als OEM-Modell (siehe Kapitel 2.2.1 „Das OEM-Modell“) können

23 Vgl. Abiteboul, Buneman, Duciu: „Data on the Web“, S. 11-17

zumindest die Knoten-Elemente alle weiteren Elemente der semistrukturierten Datensammlung referenzieren. Dies ist dadurch möglich, dass allen Elementen eine eindeutige ID zugewiesen wird. Z.B. `Fahrzeug: &o1`. Um von einem Element zu einem anderen zu referenzieren, wird ein Attribut zusammen mit einer eindeutigen ID angegeben, z. B: `Hersteller: &o2`. Alle Referenzen, die nicht dem Element selbst untergeordnete Elemente referenzieren, werden in dieser Arbeit als Quer-Referenz bezeichnet.

Weil es somit möglich ist, sich innerhalb des Graphen durch die gerichteten Kanten zyklisch zu bewegen, werden solche Datensammlungen als zyklisch bezeichnet.[24]
Ein zyklischer Graph ist im folgenden in der ssd-Notation dargestellt:

```
{Fahrzeug: &o1{Modell: "Passat"
     km-Stand: "35.600",
     Erstzulassung: „02/2007" ,
     Hersteller: &o2,
     Motor: &o3
     },

Hersteller: &o2 {Name : „Volkswagen",
     Ort: „Wolfsburg"
     Produkte: {Gebrauchtwagen : &o1,
             Motor: &o3 }

     },

Motor: &03 {Name: „OttoV2",
     Kraftstoff: „Benzin"
     Hubraum : "2.0 Liter"
     PS : „120"
     }
}
```

2.4.2 Die semistrukturierte Notation XML

Sehr weit verbreitet ist hingegen die Notation von semistrukturierten Daten mit XML, die von dem W3C-Konsortium[25] standardisiert worden ist. Diese dient als Datenaustauschformat im Internet und wird zusätzlich in vielen Applikationen als Datenablageformat verwendet.

In XML lassen sich mit folgender Notation Attribute mit sog. Tags notieren, deren Name frei festgelegt werden kann:

```
<tagname [attribut_1="wert_1"] [attribut_2="wert_2"]
```

24 Vgl. Abiteboul, Buneman, Duciu: „Data on the Web", S. 16-17
25 Quin: „Extensible Markup Language (XML)", http://www.w3.org/XML (2.2007)

```
[attribut_n="wert_n"] />
```

Der ssd-Datensatz

```
{Fahrzeug: {Modell: "Passat" } }
```

sieht in XML wie folgt aus:

```
<Fahrzeug Modell="Passat"/>
```

Ein Tag kann weitere Inhalte und/oder weitere Untertags enthalten:

```
<tagname [attribut_1="wert_1"] [attribut_2="wert_2"]
[attribut_n="wert_n"] />
     inhalt1
     <untertag_1/>

     <untertag_2/>
     ....
<tagname>
```

Somit existieren innerhalb von XML zwei Möglichkeiten, Eigenschaften von Objekten zu spezifizieren:

a) durch XML-Attribute
b) durch Untertags

Der ssd-Datensatz (s. o.) kann auch mit einem weiteren Untertag beschrieben werden:

```
<Fahrzeug>
     <Modell>
           Passat
     </Modell>
</Fahrzeug>
```

2.4.2 Document Type Definitions DTDs

Für die XML-Dokumente existiert eine weitere Notation, welche die Bezeichnung DTD (Document Type Definition) trägt. Diese Notation beschreibt die Struktur eines XML-Dokuments[26].

XML-Dateien mit DTD sind „strukturierter“ als XML-Dateien ohne DTD.
XML-Dateien ohne DTD haben keine Typisierung.

26 Abiteboul, Buneman, Duciu: „Data on the Web“, S. 38

Innerhalb eines XML-Dokumentes können Elemente bzw. Tags und deren Attribute beliebig definiert werden – ohne Einschränkungen. Es ist prinzipiell möglich, dass die DTD nur einen Teil der Elemente innerhalb des XML-Dokumentes festlegt. Mit Hilfe einer DTD kann definiert werden, welche Elemente es geben darf, und welche Attribute diese Elemente haben dürfen oder müssen; ebenso kann die Menge der möglichen Werte eingeschränkt werden. Zusätzlich kann die Menge möglicher untergeordneter Elemente mit DTDs definiert werden. Die in der DTD beschriebenen Typen können impliziert werden; ein Verfahren wird in Kapitel 2.4.4 „Ablegen von semistrukturierten Daten“ beschrieben.

Obwohl das XML-Dokument einer Strukturbeschreibung unterliegt, kann nicht von strukturierten Daten gesprochen werden.

Trotz der Möglichkeit der weiteren Strukturierung mit DTDs, befinden wir uns immer noch auf der semistrukturierten Ebene der Datenhaltung.
Dies ist damit begründet, dass strukturierte Daten aus technischer Sicht einem sog. Data Dictionary[27] unterliegen, der die Struktur der Daten beschreibt.

Zur Struktur der Entities gehören u. a. die Beziehungen, Attribute und Werte mit ihren Datentypen. Ein Zugriff auf die abgelegten Daten ohne Data Dictionary ist nicht möglich.

Anders ist es bei semistrukturierten Daten, die grundsätzlich wie eine Textdatei aufgebaut sind.
Auch sind die Werte der Attribute nicht mit Datenstrukturvorgaben wie String, Integer, Float, Date, Number etc. definiert, sondern werden grundsätzlich als Zeichenketten (Strings) dargestellt.

Somit kann eine XML-Datei, die mit einer DTD validiert wurde, unabhängig von der DTD bearbeitet und verändert werden.
Verschiedene XML-Dateien, die wiederum mit ein- und derselben DTD validiert werden können, gehören damit einer gleichen Äquivalenzklasse an.

Da die Struktur der DTD von den verarbeitenden Algorithmen abgeleitet wird, können semistrukturierte Daten in XML mit DTD nur von einem Programm in einer Version erzeugt und auch mit einem Programm und einer Version weiterverarbeitet werden - es sei denn, bei der Weiterverarbeitung werden semantisch orientierte Abfragen oder Verarbeitungsmethoden eingesetzt (siehe Kapitel 2.4.5 „Abfragen von semistrukturierten Daten“).
Möglicherweise können DTDs auch durch Typenerkennungsverfahren, wie Simulation (Abiteboul) erzeugt werden, da mit diesem Verfahren Typen von Objekten „Klassen“ erkannt werden.
Programmänderungen - wie hier im Analysesystem - führen auch zur Anpassung der DTD.

Zusätzlich bietet die semistrukturierte Konzeption die Möglichkeit, dass Elemente, die in diesem Fall Wörter und Satzphrasen beschreiben, beliebig aufeinanderfolgen können.
Die DTD-Notation bietet Parameter Entities[28] an, die eine beliebige Reihenfolge und

27 Sauer: „Relationale Datenbanken“ S. 25
28 Münz, Kapitel: „Parameter Entities für komplexere DTDs“,

Anzahl von Unterelementen eines übergeordneten Elementes ermöglichen.
Dies ist bei der strukturierten ER-Modellierung nicht auf direktem Wege möglich.

2.4.3 Ablegen von semistrukturierten Daten

Um zu zeigen, wie Daten abgefragt werden können, denen kein Strukturmodell zugrunde liegt, soll anhand eines Beispiels eine OEM-Datenbank gefüllt werden. Die grundlegenden Quellen sind zwei unterschiedliche, strukturierte Datenbanken *S1* und *S2*.

```
O1:
Fahrzeug
Marke: VW
Modell: „Passat“
km-Stand: 38.350
Preis: 12.000
EZ: 12.03
```

```
O2:
Fahrzeug
Laufleistung: 34.570
Modell: „Golf“
Preis: 15,800
Erstzulassung: 12/2004

O3:
Hersteller
Name: Volkswagen
```

O1 ist ein Objekt aus Datenbank *S1, O2* und *O3* stammen aus *S2*. Beide Fahrzeug-Objekte *O1* und *O2* zusammen mit *O3* repräsentieren Fahrzeuge mit bestimmten Eigenschaften, die zwar unterschiedlich definiert aber semantisch identisch sind.

O1, *O2* und *O3* können in eine semistrukturierte Datensammlung übernommen werden, ohne dass die Modellstruktur der Objekte in irgendeiner Weise transformiert werden muss, da die unterschiedliche Typdefinition dieser Objekte für semistrukturierte Daten irrelevant ist.

Gehen wir davon aus, dass wir zwei verschiedene, strukturierte Datenbanken für Fahrzeuginformationen haben: *S1* und *S2*.

S1 enthält die Typbeschreibung *T1*, die dem Objekt *O1* entspricht, das folgendermaßen spezifiziert ist:

http://de.selfhtml.org/xml/dtd/entities.htm#parameter_entities (2.2007)

T1:

Fahrzeug

ID	Number
Marke	String
km-Stand	Integer
Preis	Integer
EZ	Datum mm.yy

S2 enthält die Typbeschreibungen *T2* und *T3*, die den Objekten *O2* und *O3* entsprechen:

T2:

Fahrzeug

ID	Number
Hersteller	Referenz ID zu Hersteller
Laufleistung	Integer
Preis	Integer
Erstzulassung	Datum

T3:

Hersteller

ID	Number
Name	String

Es wird der Dateninhalt von *S1* und *S2* in eine semistrukturierte Datenbank *S3* geschrieben.

Die Objekte liegen nun in *S3*, jedoch die Strukturen *T1*, *T2* und *T3* liegen nicht vor. Diese Typbeschreibungen müssen impliziert werden. Typbeschreibungen können jedoch durch das häufige Vorkommen aller Objekte, die ursprünglich *T1* unterliegen, impliziert werden. Ein mögliches Verfahren zur Implikation von Datentypen ist die Simulation, in der die Eigenschaften der Objekte, die in einer semistrukturierten Datenbank vollständig mit Referenzen wiedergegeben werden, miteinander verglichen werden.

Simulation:

$G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ sind zwei gerichtete Graphen.

Es gibt eine Menge von Attributen *L*, die für beide Graphen gilt, also eine Vereinigungsmenge $L = (A_1 \cup A_2)$ der Attribute Mengen A_1 und A_1 für die Graphen G_1.
Eine Beziehung zwischen zwei Knoten wird folgendermaßen ausgedrückt: $x_1[l]y_1$

Die Relation von R von V_1 und V_2 ist erfüllt, wenn folgende Bedingung gilt:

$$\forall l \in L \forall x_1, y_1 \in V_1 \forall x_2 \in V_2 (x_1[l]y_1 \Lambda x_1 R x_2 \Rightarrow \exists y_2 \in V_2 (x_1 R y_2 \Lambda x_2[l]y_2))$$

Das heißt, für alle Beziehungen innerhalb des Graphen V_1 mit x_1 und y_1 aus der Menge der Knoten des Graphen existiert ein y_2 aus der Menge der Knoten des Graphen V_2, das die gleichen Attribute wie y_1 hat. [29]

Dieses Verfahren kann für die Erkennung der Typen - wie hier im obigen Beispiel *T1'*, *T2'* und *T3'* - angewandt werden.
Im folgenden soll gezeigt werden, wie die Typen *T1, T2* und *T3,* die in den Datenbankmodellen *S1* und *S2* enthalten sind, in *T1',T2'* und *T3*' transformiert werden (*T1, T2, T3) → (T1', T2', T3')*, indem diese aus den Objekten *O1, O2* und *O3* impliziert werden:

Sowohl G_1 als auch G_2 repräsentieren den gerichteten Graphen aus den semistrukturierten Dateien *S3*.

Es wird für jedes y_1 ein x_1 Element gesucht, das eine direkte Referenz $x_1[l]y_1$ aufweist. Es wird nun ein y_2 aus demselben Graphen gesucht, das nicht gleich y_1 ist und die Bedingung der Simulation erfüllt. Ist ein y_2 gefunden, hat dies die gleichen Attribute wie y_1.

Ist Ay_1 die Menge der Attribute von y_1
und Ay_2 die Menge der Attribute von y_2,
so gilt: $Ay_1 \subseteq Ay_2$. Das heißt y_2 kann noch weitere Attribute haben, die y_1 nicht hat.

Es kann nun aus den durch die Anwendung der Simulation herausgefundenen, übereinstimmenden Attribute von y_1 und y_2 ein Datentyp T definiert werden.

Auf das Beispiel der semistrukturierten Daten S3 würde der Datentyp T1' folgendermaßen mit der Simulation impliziert werden können:
Der Datentyp T1' hat die Bezeichnung „Fahrzeug“ (diese wird innerhalb eines OEM-Graphen durch eine auf y_2 gerichtete Referenz ausgedrückt) und weitere ausgehende Attribute wie `Marke, km-Stand, Preis, EZ`.
Dieses Element wird mit seinen Attributen in der DTD-Notation folgendermaßen beschrieben:

29 Vgl. Abiteboul, Buneman, Duciu: „Data on the Web“, S. 132-133

T1':

```
<!ELEMENT Fahrzeug (#PCDATA)>
<!ATTLIST Fahrzeug
  Marke #REQUIRED
  km-Stand #REQUIRED
  Preis #REQUIRED
  EZ #REQUIRED
>
```

T2' und *T3'* können folgendermaßen impliziert werden:

T2':

```
<!ELEMENT Fahrzeug (#PCDATA)>
<!ATTLIST Fahrzeug (Hersteller)
  Laufleistung CDATA #IMPLIED
  Preis CDATA #IMPLIED
  Erstzulassung CDATA #IMPLIED
>
```

T3':

```
<!ELEMENT Hersteller (#PCDATA)>
<!ATTLIST Hersteller
  ID CDATA #IMPLIED
  Name CDATA #IMPLIED
>
```

Die nun implizite Typdefinition *T1'* beschreibt einen Teil der in *S3* enthaltenen Objekte und *T2'* einen weiteren Teil. Somit sind implizierte Typbeschreibungen von semistrukturierten Daten nur partiell auf semantisch identische Objekte zutreffend.

Greiner schreibt zusammenfassend: „Bei diesen kann die Struktur implizit, irregulär oder partiell sein.“ [30]
Implizit bedeutet, da keine Strukturbeschreibungen vorliegen, müssen diese anhand der Daten impliziert werden.
Irregulär bedeutet, dass sich innerhalb der Daten die Struktur verändern kann (innerhalb der semistrukturierten Daten können sich auch unstrukturierte Daten vorfinden).

Partiell bedeutet, wiedergefundene Strukturen müssen nicht auf alle Daten zutreffen, sondern treffen nur auf Teile der Objekte zu.

30 Greiner, Kapitel 1.1: „Semistrukurierte Daten“, http://dbs.uni-leipzig.de/html/seminararbeiten/semSS99/arbeit3/kapitel1.html (2.2007)

2.4.4 Abfragen von semistrukturierten Daten

Um Abfragen (Query-Statements) für strukturierte Daten zu erstellen, wird ein Datenmodell benötigt. Dies liegt jedoch bei einem semistrukturierten Modell nicht vor.

Da wir nicht wissen, wie die Datentypen und die Attribute der Datentypen wirklich benannt sind, ist die Abfrage von semistrukturierten Daten auf direktem Wege, wie bei strukturierten Daten, nicht möglich.

Soll z. B. nach Fahrzeugen des Herstellers Volkswagen gesucht werden, die einen Kilometerstand von unter 60.000 km und eine Erstzulassung ab Januar 2007 haben, wissen wir nicht, ob wir nach Attribut „Marke“ suchen sollen oder nach einer Referenz zum Hersteller mit einem Namen. Ähnliches gilt für den Kilometerstand: hier sind verschiedene Synonyme möglich: wie km, km-Stand, Kilometerstand, Kilometer oder Laufleistung etc..

Mit Hilfe eines Select-Statements, wie es bei strukturierten Daten üblich ist, auf das semistrukturierte Datenbank-System *S3* zuzugreifen, würde mit großer Wahrscheinlichkeit keine Treffer erzielen. Wenn das generierte Select-Statement zufällig *T1'* oder *{T2', T3'}* entsprechen sollte, würde nur ein Teil der möglichen Zielmenge erfasst werden.

2.4.5 Das semistrukturierte Datenbankmanagement-System Lore

Für semistrukturierte Daten existiert heute im Open-Source-Bereich ein Datenbankmanagement-System namens Lore (Lightweight Object Repository).
Lore stellt eine eigene Abfragesprache Lorel (Lore Language) zur Verfügung, die stark an den Standard OQL angelehnt ist. [31] Weitere vom W3C-Konsortium verabschiedete Abfragesprachen für semistrukturierte Daten sind XPATH[32] und XQuery[33].

Im folgenden sollen Möglichkeiten beschrieben werden, mit der semistrukturierten Abfragesprache auf den Datenbestand der Datenbank *S3* zuzugreifen.

2.4.6 Abfragen von semistrukturierten Daten auf Lore

In unserm Beispiel sollen die Modellnamen aller Fahrzeuge mit einem Kilometerstand von unter 60.000 km gesucht werden.

Ein Select-Statement, das semantisch auch einer SQL/OQL Abfrage entspricht, würde folgendermaßen aussehen:

31 Vgl. Abiteboul, Buneman, Duciu: „Data on the Web“, S. 62
32 Clark, DeRose: „XML Path Language (Xpath) Version 1.0“, http://www.w3.org/TR/xpath (2.2007)
33 Boag: „XQuery 1.0: An XML Query Language“, http://www.w3.org/TR/xquery/ (2.2007)

```
q1:
  select F.Modell
  from Fahrzeug F
  where F.Kilometerstand < 60.000
```

Wie schon erwähnt, haben wir innerhalb unserer semistrukturierten Datenbank S3 das Problem, dass die Attribute (wie Kilometerstand) mit unterschiedlichen Synonymen beschrieben sein können.

Lorel bietet die Möglichkeit, die Pfadangaben des Select-Statements mit Patterns (reguläre Ausdrücke) zu beschreiben und verwendet folgende Notationen:

```
s1.s2  s2 ist unterhalb s1
s1|s2 : für die Disjunktion s1 oder s2
(s1)? : für ein optionales Attribut
(s1)+ : für die mindestens einmalige Wiederholung von s1
(s1)* : für die keinmalige oder beliebig häufige Wiederholung von
s1 [34]
```

Mit Hilfe der Disjunktion kann das Select-Statement q1 z. B. sämtliche Synonyme für Kilometerstand enthalten.

```
q2:
  select F.Modell
  from Fahrzeug F
  where F.(km|km-Stand|Kilometerstand|Kilometer|Laufleistung) <
60.000
```

Dieser Ausdruck kann auch mit Hilfe weiterer Notationen abgekürzt werden:

```
q3:
  select F.Modell
  from Fahrzeug F
  where F.(km(stand)?| kilometer(stand)?|Laufleistung) < 60.000
```

Lorel stellt ein Wildcard „%“ zur Verfügung, die folgendermaßen eingesetzt werden kann:

```
q4:
  select F.Modell
  from Fahrzeug F
  where F.(km(%)?|kilometer(%)?|Laufleistung) < 60.000
```

34 Greiner. Kapitel 5.1.1: „Abfrage semistrukturierter Daten“ http://dbs.uni-leipzig.de/html/seminararbeiten/semSS99/arbeit3/kapitel5.html (2.2007)

Diese Abfrage wollen wir nun auf alle Fahrzeuge vom Hersteller „Volkswagen“ beschränken.

Dabei tritt folgende Problemstellung auf:
Bei *T1* ist der Hersteller mit Hilfe des Attributs „Marke“ angegeben, während bei *T2* der Hersteller durch den weiteren Typ *T3* repräsentiert wird. *T3* enthält wiederum das Attribut „Name“. Somit sind hier zwei Pfade möglich, um an den Hersteller zu gelangen:

```
Fahrzeug.Marke
```

oder

```
Fahrzeug.Hersteller.Name
```

Ein regulärer Ausdruck, um beide Pfade erfassen zu können, würde folgendermaßen aussehen:

```
    Fahrzeug(.Hersteller)?.(Name|Marke)
```

Somit erweitern wir den Ausdruck q4

```
q5:
  select F.Modell
  from Farhrzeug F
  where ((F.(km(%)?|kilometer(%)?|Laufleistung) < 60.000 ) and
    F(.Hersteller)?.(Name|Marke) = „Volkswagen“)
```

Dieses Kapitel hat die Möglichkeiten gezeigt, mit der semistrukturierten Abfragesprache Lorel ein Statement so zu erweitern, dass Fahrzeugobjekte unabhängig von der Typdefinition erkannt werden können, die semantisch bestimmte Eigenschaften aufweisen.

Das folgende Kapitel „Einschränkungen“ soll jedoch auch Aspekte beleuchten, die dieses Verfahren kritisch betrachten.

2.4.7 Einschränkungen

Im allgemeinen besteht die Idealvorstellung, Abfragen definieren zu können, ohne sich vorher im klaren zu sein, wie die Objekt-Typen in der Datenbank definiert sind. Abfragen müssen in der Lage sein, semantisch übereinstimmende Typen zu erkennen und dementsprechend deren Merkmale nach den Abfragebedingungen zu untersuchen.

Abfragen, wie im Beispiel des vorigen Unterkapitels beschrieben, können dazu führen, dass Objekte aus der Abfragemenge ausgeschlossen sind, die jedoch in diese hineingehören, oder dass Objekte gefunden werden, die vom Anwender nicht erwünscht sind.
Einige Ursachen für Fehl-Erkennungen:

- Objekte oder deren Eigenschaften werden mit einem unbekannten Synonym oder Kürzel referenziert. Z.B. `FZ` für `Fahrzeug` oder `Laufleistung(km)` für `Kilometerstand`.

- Objekte haben Eigenschaften, die vom Anwender nicht erwünscht sind, z.B. ein Fahrzeug ist defekt oder nicht mehr vorhanden etc..

- Es liegen Objekte vor, die wiederum andere Objekte beschreiben. Z.B. befinden sich in der Datenbank auch Fahrzeugscheine, die wiederum Fahrzeuge beschreiben. Die Fahrzeuge selber sind jedoch nicht für den Anwender verfügbar.

- Andere Objekte werden mit gleichen Referenzen beschrieben; z. B. ein PKW-Transporter wird als Fahrzeug in der Datenbank abgelegt.

- Werte von Attributen enthalten wiederum eigenständige Notationen, wie z. B. unterschiedliche Datenformate oder Zahlenformate. Sie können eigenständige Notationen enthalten oder Fließtexte, die wiederum mehrere Attribute-Werte-Paare enthalten. Ein Konzept zur Extraktion verschiedener Datumsformate und Fließtexte stellt in dieser Masterarbeit die beschriebene Softwarelösung IPEE dar.

Somit ist bei einer semistrukturierten Abfrage weder 100% sicher gestellt, dass alle semantisch übereinstimmenden Elemente gefunden werden, noch ist sicher gestellt, ob in der Treffermenge nicht semantisch übereinstimmende oder nicht erwünschte Elemente vorhanden sind.

Fazit:

Die Anwendung von Abfrage auf semistrukturierte Daten eignet sich in der Praxis nur in Einsatzbereichen, in denen auf die Anforderung der exakten Datenabfrage verzichtet werden kann und deren erfasste Daten Fehlerquoten aufweisen dürfen.
Einsatzgebiete, die mit dieser Einschränkung auskommen können, sind heute eher noch auf Bereiche wie künstliche Intelligenz beschränkt. In der Praxis des operativen Geschäftes haben sich semistrukturierte Datenbanksysteme bis heute nicht durchgesetzt.

2.5 Gegenüberstellung

Als Abschluss des Kapitels 2 „Konzeptionen von Datenstrukturen“ sollen die Eigenschaften und Anwendungsbereiche der drei unterschiedlichen Konzeptionen tabellarisch zusammengefasst werden:

	strukturierte Daten	**semistrukturierte Daten**	**nicht strukturierte Daten**
Strukturmodelle	liegen grundsätzlich vor	liegen nicht vor, können aber impliziert werden	liegen nicht vor, können auch nicht impliziert werden
Eigenschaften der Daten	sind streng nach einem Typenmodell spezifiziert	auftretende Strukturen sind partiell und irregulär	Strukturen sind nicht auffindbar
Modelle	ER-Modell	OEM-Modell	Daten unterliegen nur ihren für das Dateiformat spezifizierten Protokollen
Notationen/ Sprachen	innerhalb ER-Diagramme	ssd XML mit DTD XML ohne DTD	
Abfragesprachen	SQL/OQL	Lorel, XPath, XQuery	keine standardisierten Abfragesprachen
heute erhältliche DBMS (Datenbank-Management-Systeme)	Oracle, DB2, MS SQL-Server, Informix, MySql, PostGre SQL, ...	Lore	Daten werden generell als Dateien im Dateisystem des Betriebssystems abgelegt. Sie können jedoch auch als Blobs[35] RDBMS abgelegt werden.
Verwendung	IT-Systeme innerhalb aller Branchen	künstliche Intelligenz, Sprachanalyse	Anwenderprogramme, die festgelegte Protokolle verarbeiten können

Tabelle 1: Zusammenfassung der Konzeptionen von Datenstrukturen

35 binary large objects

3 Wort-Analyse

Nachdem im vergangenen Kapitel die Konzeption der semistrukturierten Daten beschrieben wurde, soll nun auf die Wort- und Phrasen-Analyse für Ausdrücke der deutschen Sprache eingegangen werden; dies jedoch zunächst auf der funktionalen Ebene.
Da die Realisierung auf der technischen Ebene sehr stark von meiner Softwarelösung IPEE unterstützt wird, sollen Pattern-Extraktions-Verfahren, die dem Erkennen von Wörtern und Endungen und zur Phrasen-Analyse dienen, in Kapitel 11 „ Semantisch orientierte Verarbeitung“ beschrieben werden.
Im Kapitel 5 „Anwendung der Konzeption semistrukturierter Daten für Satzmuster“, das nach der Wort- und Phrasen-Analyse folgt, soll darauf eingegangen werden, wie die Konzeption der semistrukturierten Daten auf die Wort- und Phrasen-Analyse angewandt wird.

Dieses Kapitel soll kurz gehalten werden. Es soll hier auf weitere Diplomarbeiten, die innerhalb des Bereiches Informatik des Instituts für Nachrichtentechnik der Fakultät 07 der FH Köln durchgeführt wurden, verwiesen und nur relevante Unterschiede dokumentiert werden.

In den Diplomarbeiten von Dischert und Kiefel[36] und von Bernd Michael Johann[37] wurden Analyse-systeme zu Wort- und Satzphrasen auf Basis eines relationalen (somit strukturierten) Datenbank-systems erstellt. Die Verarbeitung und Ausgabe mit strukturierten Daten soll im Kapitel 5 „Anwendung der Konzeption semistrukturierter Daten für Satzmuster“ mit der Verarbeitung von semistrukturierten Daten in Beziehung gesetzt werden.

Für das Verständnis des Wort-Analyse-Verfahrens ist es sinnvoll, Wortarten in einfache und flektierte Wortarten aufzuteilen.

3.1 Einfache Wörter

Dazu gehören unflektierbare[38] Wortarten. Die Erkennung erfolgt durch das Vergleichen des Wortes mit festen Einträgen in Datensätzen, die möglichst alle Wörter einer bestimmten einfachen Wortart enthalten. Die Qualität der Analyse ist von der Richtigkeit und Vollständigkeit der Datensätze abhängig.
Wortarten wie Numerale, Präpositionen, Konjunktionen, Pronomen, Adverbien und Hilfsverben werden im Analyseverfahren nacheinander abgearbeitet.[39] Die Datensätze

36 Dischert/Kiefel, S. 42-61
37 Johann, S. 24-42
38 Wörter, die weder konjugiert noch dekliniert werden
39 Dischert/Kiefel, S. 43

enthalten entsprechend die notwendigen Attribut-Werte-Tupels, die das Wort beschreiben. Das Wort „er“ z. B. ist ein Personalpronomen und hat noch zusätzlich folgende Attribut-Werte-Tupels:

```
Wort: "er" Kasus: "Nom.", Person: "3.Pers." Numerus: "Sing."
```

Auf die Verarbeitung der Wort-Dateien und ihre Formatierung wird auch in Kapitel 10.1 „Einladen der Wortdateien“ Bezug genommen. In der entsprechenden Datei „`Pronomen.ssd`“ (im Verzeichnis `WordData/`) ist folgender Eintrag wiederzufinden:

```
Personalpronomen: {Wort: "er", Kasus: "Nom.", Person: "3.Pers."
Numerus: "Sing.", Genus : "Mask." },
```

Es existieren auch einfache Wörter, die flektiert sind. In diesem Fall ist für jede Beugung ein Eintrag in den entsprechenden Datensätzen vorhanden, wie z. B. das Wort „mein“, zu dem es die Flexionen „meine“, „meinen“, „meinem“, „meines“, „meiner“ gibt.

Für die Erkennung von Adverbien, Hilfsverben, Adjektiven und Präpositionen wurde aus den vorhandenen Datensätzen von Dischert und Kiefel Gebrauch gemacht. Diese sind im Verzeichnis `VerbSemantik/Nebsy/` wiederzufinden (Dateiendung ist `.DAT`).

Für Numerale, Konjunktionen, Pronomen, Satzzeichen und Abkürzungen wurden innerhalb der Entwicklung dieses Analysesystems eigene Dateien angelegt, da in den entsprechenden Dateien aus dem Projekt Nebsy bestimmte notwendige bzw. sinnvolle Attribute nicht vorhanden sind.
Artikel sind ebenfalls innerhalb der Pronomen abgelegt. Diese Dateien sind im Verzeichnis `WordData/` abgelegt und sind in der ssd-Notation beschrieben (Dateiendung ist `.ssd`).

3.2 Flektierte Wörter

Bei Verben, Substantiven, Adjektiven und Partizipien werden bestimmte Werte von Attributen durch die Endungen (Flexionen) erkannt.

3.2.1 Partizipien

Auch die Analyse der Partizipien lehnt sich an das Verfahren von Dischert und Kiefel[40] an. Dieses Verfahren erkennt alle Partizipien innerhalb der Wort-Analyse als Adjektive.
Ein Partizip kann jedoch sowohl als Adjektiv (nähere Beschreibung eines Substantivs) als auch als Adverb (nähere Beschreibung eines Verbs) dienen:

40 Dischert/Kiefel, S. 47

Der gehende Mann schaut sich um. (Partizip als Adjektiv)
Der Mann schaut sich gehend um. (Partizip als Adverb)
Deswegen werden in dem von mir erarbeiteten Wort-Analyse-System die als Partizip erkannten Wörter auch als Partizip ausgegeben und nicht als Adjektiv (bzw. Adverb).

3.2.2 Adjektive

Die Erkennung von Adjektiven lehnt sich sehr stark funktionell an das von Bernd Michael Johann entwickelte Analyseverfahren[41] an. Es wird ebenfalls die Datei `Adj.DAT` aus dem Projekt von Dischert und Kiefel[42] (Nebsy Projekt) verwendet.
Johann beschreibt einige nicht steigerbare Adjektive wie „tot“ oder „schwanger“.[43] Aus der verwendeten Datei `Adj.DAT` ist jedoch nicht erkennbar, welche Adjektive nicht steigerbar sind. Die Analyse-Verfahren erkennen z. B. toter und totest als gesteigertes Adjektiv; d. h. eine 100%ige Aussage über die Existenz eines gesteigerten Wortes kann nicht gemacht werden.

3.2.3 Verben

Als sehr aufwändig erwies sich ein eigenes Verfahren zur Erkennung der Verben, das sich in zwei wesentliche Komponenten aufteilt, die nacheinander ablaufen („Hybrid“-Lösung). Die erste Komponente ist verantwortlich für die Erkennung unregelmäßiger Verben[44] mit Hilfe der `Verb.DAT` Datei aus dem Nebsy-Projekt. Die zweite Komponente erkennt nur die schwachen/ regelmäßigen Verben.

Anders als bei den Lösungen von Dischert/Kiefel und Johann[45], wird die Existenz des möglichen erkannten Verbs mit dem Internet-Lexikon wissen.de geprüft.
Weitere Unterschiede:
Unregelmäßige Verben können in der 2. und 3. Person Singular vom Wortstamm der Grundform (1 Spalte aus `Verb.DAT` Datei) oder aus dem Stamm der 2. Spalte abgeleitet sein, z. B.:

41 Johann S. 28-30
42 Dischert/Kiefel
43 Johann S. 28
44 sehr ähnlich mit starken Verben
45 Dischert/Kiefel S. 56-60 / Johann S. 31

1. Person Sing. Präsens	*2. Person Sing. Präsens*	*3. Person Sing. Präsens*	*Stamm Grundform*	*Stamm Präsens*
ich helfe (G)	du hilfst (3)	er/sie/es hilft (3)	helf	hilf
ich gebäre (G)	du gebierst (3) / du gebärst (G)	er/sie/es gebiert(3) er/sie/es gebärt (G)	gebär	gebier

Tabelle 2: Ableitung von unregelmäßigen Verben

Somit besteht keine feste Regel, ob nun die 2. oder 3. Person Singular vom Stamm der Grundform abgeleitet ist oder vom Stamm der 3. Person Präsens.

Um alle starken Verben[46] in der 2. und 3. Person Singular zu erkennen, versucht das für diese Masterarbeit entwickelte Analysesystem sowohl den Stamm der Grundform als auch den Stamm der
Präsensform mit dem Stamm des zu erkennenden Verbs zu vergleichen.
Dieses tolerante Verfahren führt jedoch auch dazu, dass falsch gebeugte Verben wie „du helfst“ als Verb „helfen“ in 2. Person Singular erkannt werden.

Sollte das zu analysierende Wort mit der unregelmäßigen Verberkennung nicht erkannt werden, folgt nun die Ausführung der zweiten Komponente, die für die Erkennung regelmäßiger Verben verantwortlich ist.

Durch Patterns (siehe Kapitel 10 „Implementiertechnische Aspekte der Wort- und Phrasen-Analyse“) werden Endungen erkannt und der Stamm herausgeschnitten. An dem heraus-geschnittenen Stamm werden die Endungen „en“ oder „n“ angehängt. Darauf erfolgt eine Abfrage auf das Web-Lexikon wissen.de.[47] Es wird hier geprüft, ob das gebildete Wort in der Grundform existiert.

Es kann jedoch vorkommen, dass bestimmte eingedeutschte Verben nicht erkannt werden, weil sie im Internet-Lexikon nicht vorhanden sind, wobei es überrascht, dass wissen.de Verben wie „klicken“, „linken“, „googeln“ und „simsen“ kennt.

Das hier vorgestellte, grundlegende Verfahren hat dadurch an Komplexität gewonnen, dass auch evtl. angehängte Präfixe oder auch Perfektformen wie z. B. „zusammengekommen“ erkannt werden.

46 Starke Verben sind unregelmäßigen Verben sehr ähnlich. Diese zeichnen sich durch den Wechsel des Stammvokals bei der Konjugation aus.

47 Wörterlexikon im Internet. Enthält Bertelsmann Wörterbuch (Wörterbuch der Deutschen Sprache) und Wahrig Deutsches Wörterbuch

Imperative:

Bei beiden Verfahren werden auch Imperative erkannt:

geh:

```
Verb: {Wort: "gehen", Person: "Imperativ Sing.", Tempus:
"Praesens ", }
```

gehe:

```
Verb: {Wort: "gehen", Person: "1. Pers. Sing./Imperativ Sing.",
Tempus:
"Praesens", }
```

geht:

```
Verb: {Wort: "gehen", Person: "3. Pers. Sing./2. Pers.
Plu./Imperativ Plu.", Tempus: "Praesens", }
```

Präfixe:

Auch werden bei der Erkennung der Verben Präfixe abgetrennt, wie z. B. bei dem Wort „zusammenbringen“:

```
Verb: {Wort: "bringen", Person: "1. Pers. Plu./3. Pers.
Plu./Grundform", Tempus: "Praesens", Praefix: "zusammen", }
```

Die 1. und 3. Person Plural ist auszuschließen, da im Satz das Präfix vom Verb getrennt ist.

3.2.4 Substantive

Nicht übernommen wurde die Erkennung von Substantiven mit Hilfe der Datei `Subst.DAT` aus dem Nebsy-Projekt, da diese Datei auf 1300 Substantive begrenzt ist und die Richtigkeit nicht sicher gestellt ist. Auch innerhalb der Diplomarbeit von Bernd Micheal Johann wurde diese Datei stark bereinigt.[48]

Es wurde versucht, eine Substantiv-Erkennung mit dem Weblexikon wissen.de zu entwickeln. Jedoch zeigt sich mit diesem Verfahren eine zu geringe Erkennungsquote. Deshalb wurde eine weitere Internet-Ressource herangezogen und zwar das Wörterbuch canoonet[49].
Dieses Lexikon ist in der Lage, flektierte Wörter - eben auch Substantive (wie Fahrräder) -

48 Johann, S. 33
49 Wörterlexikon im Internet von der Firma Canoo Engineering AG

zu erkennen und die Grundform auszugeben. (Das ist mit wissen.de nicht möglich.) Zusätzlich bietet es bei einem gefundenen Wort Flexions-Tabellen an, mit deren Hilfe bei Substantiven Kasi und Numeri erkannt werden können.

Prinzipiell wäre es möglich, auch weitere Wortarten mit dem Lexikon canoonet zu erkennen. Dies würde jedoch rechtliche Fragen aufwerfen. Für den alleinigen Zweck der Substantiv-Analyse konnte von canoonet eine Genehmigung eingeholt werden. Innerhalb dieser Masterarbeit wurde es daher bevorzugt, eigene Lösungswege (z. B. für das Erkennen von Verben) auszuarbeiten.

4 Phrasen-Analyse

Innerhalb der semantischen Verarbeitung von Sätzen der deutschen Sprache spielt die grammatische Erkennung bzw. Verarbeitung von Satzteilen oder Phrasen eine bedeutende Rolle.
Der Grammatikwissenschaftler Fillmore entwickelte ein sog. „Case Grammar Model".

In den natürlichen Sprachen existieren Kasi wie Nominativ, Genitiv, Dativ und Akkusativ, im folgenden als Oberflächen-Kasus bezeichnet. Fillmore stellt einen sogenannten Tiefen-Kasus auf, der die Fälle Handelnder, Instrument, Erleidender, Ort und Ziel auf semantischer Ebene unterscheidet.

Der Versuch Fillmores ist, durch Regeln und Erkennen von syntaktischen Funktionen vom Oberflächen-Kasus auf den Tiefen-Kasus zu schließen. Jedoch ist dies nur innerhalb von Fallbeispielen möglich.

Solche syntaktischen Funktionen können durch die Reihenfolge der Wörter – z. B. durch das Subjekt und direkte Objekt, die Deklination in Kasi der Substantive und durch Präpositional-phrasen - gekennzeichnet sein, auf die nachfolgend eingegangen wird. [50]

Das Spektrum verschiedener Phrasen und Satzmuster ist nahezu unerschöpflich. Deswegen wurde es bevorzugt, sich auf sehr häufig angewandte Nominal- und Präpositionalphrasen zu beschränken.

4.1 Nominalphrasen

„Nominalphrasen sind zusammenhängende Satz-Elemente, die das Subjekt oder Objekt bilden."[51] Dischert und Kiefel beschreiben sämtliche Masken[52] zur Erkennung von Nominalphrasen innerhalb ihres Regelwerks.

Davon sind folgende Regeln übernommen:
(Siehe Tabelle nächste Seite)

50 Vgl. Fillmore: „Fillmore's case grammar", S. 6
51 Dischert/Kiefel, S. 68
52 Dischert/Kiefel, S. 62 - 64

Muster	Beispiel
|[Art~] [Zahl] [n*Adj] [Subst]|Block|10|	Die drei schönen Tassen Die drei schönen bunten Tassen
|[Zahl] [n*Adj] [Subst]|Block|12|	drei schöne Tassen drei schöne bunte Tassen
|[Art~] [n*Adj] [Subst]	Die schönen Tassen Die schönen bunten Tassen
|[Pro~] [n*Adj] [Subst]|Block|18|	Diese schönen Tassen (Demonstrativpronomen) Meine schönen Tassen (Possesivpronomen)
|[Adj] [Satzz(,)] [Adj] [Subst]|Block|22|	schöne, bunte Tassen
|[Adj] [Konj] [Adj] [Subst]|Block|24|	schöne und bunte Tassen
|[Adj] [Subst]|Block|28|	schöne Tassen
|[Art~] [Zahlw] [Subst]|Block|30|	Die drei Tassen
|[Art~] [Zahl] [Subst]|Block|34|	Die 40 Tassen
|[Zahlw] [Subst]|Block|32|	drei Tassen
|[Zahl] [Subst]|Block|32|	40 Tassen
|[Subst]|Block|38|	Tassen

Tabelle 3: Verwendete Masken aus der Diplomarbeit von Dischert und Kiefel

Innerhalb der Diplomarbeit von Dischert und Kiefel werden Partizipien auch wie Adjektive behandelt. Es erkennt zusätzlich folgende:

Muster	Beispiel
|[Art~] [Zahl] [n*Adj] [Subst]|Block|10|	die drei lachenden Männer
|[Zahl] [n*Adj] [Subst]|Block|12|	drei lachende Männer
...	

Tabelle 4: Verwendete Masken aus der Diplomarbeit von Dischert und Kiefel mit Adjektiven

.

Wie in Kapitel 3.2.1 „Partizipien“ erwähnt, werden in dieser Masterarbeit im ausgearbeiteten Analysesystem Adjektive und Partizipien auseinander gehalten.

Das ausgearbeitete Phrasen-Analyse-Muster bzw. Pattern (siehe Kapitel 10.5 „Erkennung von Nominalphrasen“) erkennt parallel zu den Adjektiven auch Partizipien.

4.2 Präpositionalphrasen

Fillmore sagt, dass auch durch die Präposition der Präpositionalphrase eine semantisch übertragbare Aussage gemacht werden kann.[53]
Deshalb ist auf diese Art von Phrasen innerhalb der semantisch orientierten Verarbeitung ein besonderes Augenmerk zu legen.

Im Deutschen gibt die Präposition lediglich Auskunft darüber, ob die gesamte Präpositionalphrase instrumental, temporal, lokal, kausal oder modal zu verstehen ist. Eine Präposition kann auch mehrere dieser Modi ausdrücken, z. B. kann die Präposition „von" temporal, lokal, kausal oder modal sein.

„Präpositionen verlangen immer eine eingebettete Phrase als Ergänzung und bilden mit dieser zusammen eine Präpositionalphrase."[54] Die eingebettete Phrase ist sehr häufig eine Nominalphrase, es sind jedoch auch Adjektivphrasen[55] bzw. Adverbphrasen möglich. Adjektivphrasen bestehen sehr häufig selber nur aus dem Adjektiv; sie sind sozusagen 1-Wort-Phrasen[56]. Ähnlich verhält es sich bei den Adverbphrasen.[57]

Das hier erarbeitete Analyse-System ist deswegen erstmals für die Erkennung von Präpositionalphrasen ausgelegt, bei denen nach einer Präposition eine Nominalphrase folgt.

Im folgenden Kapitel ist anhand eines Beispiels beschrieben, wie eine erkannte Präpositionalphrase, in der eine Nominalphrase eingebettet ist, in einer semistrukturierten Notation dargestellt wird.

53 Fillmore, Charles: „Fillmore's case grammar", S. 6
54 Eisenberg: „Duden Band 4, Die Grammatik", Randmarke 1297
55 Eisenberg: „Duden Band 4, Die Grammatik", Randmarke 1283/ Randmarke 1290
56 Eisenberg: „Duden Band 4, Die Grammatik", Randmarke 1169
57 Eisenberg: „Duden Band 4, Die Grammatik", Randmarke 1290 – 1296

5 Anwendung der Konzeption semistrukturierter Daten für Satzmuster

Im vorigen Kapitel wurde die Wort- und Phrasen-Analyse auf funktionaler Ebene beschrieben. Anders als bei der Diplomarbeit von Dischert/Kiefel und Johann, erfolgt die Ausgabe der Daten nicht auf strukturierter Ebene, sondern auch auf semistrukturierter Ebene.

Innerhalb dieses Kapitels soll die Frage geklärt werden, wie die Konzeption der semistrukturierten Daten eingesetzt werden kann, und welche Unterschiede die Verarbeitung mit semistrukturierten Daten gegenüber strukturierten Daten aufweist.
In Kapitel 2 „Konzeptionen von Datenstrukturen“ wurde der Begriff der semistrukturierten Daten definiert und nachfolgend auch beschrieben, wie diese mit Hilfe der Notationen ssd oder XML wohlgeformt beschrieben bzw. abgelegt werden können.

Im praktisch realisierten Analyseprozess durchlaufen die semistrukturierten Daten, die anfangs Sätze der deutschen Sprache sind, die verschiedenen Stadien „nicht geformt“, „wohlgeformt“ und mit Dokument-Type-Definitionen DTD „strukturiert“. Diese drei Stufen sollen in diesem Kapitel nochmals anhand dieser Anwendung unterschieden werden.

Nach Definition von semistrukturierten Daten in Kapitel 2.3 „Nicht strukturierte Daten“ kann ein textuell (Ascii-Format) dargereichter deutscher Satz durchaus als semistrukturiert bezeichnet werden: Er ist semistrukturiert, wenn für den Betrachter oder für das verarbeitende System ganz oder teilweise eine OEM-Struktur wieder zu erkennen ist. Kandidaten für OEM-Strukturen innerhalb von Sätzen der natürlichen Sprache sind bekannte Wörter, denen Wortarten zugewiesen werden können, Endungen, Phrasen und (grammatische) Satzstrukturen.

Z. B. soll ein natürlicher Satz der deutschen Sprache, wie „*Ich fahre mit dem Auto in die Stadt*“, textuell in das Analysesystem eingegeben werden. Es wird im folgenden beschrieben werden, wie der Beispielsatz in ein OEM-Modell transformiert wird. Das OEM-Modell wird im ssd-Format ausgegeben. Zuerst erfolgt eine Analyse der Wortarten, die sowohl einfache (nicht flektierte) Wörter erkennt und gleichzeitig flektierte Wörter lemmatisiert[58], d. h. die Endungen erkennt.

Grundsätzlich wird für jedes Wort ein Element erzeugt, welches die Wortart als Elementnamen (Bezeichnung) trägt; zusätzlich enthält es das Attribut „Wort“, in das die Grundform des Wortes wiedergegeben wird, und anschließend weitere Attribute, wie „Kasus“, „Numerus“, „Genus“, „Tempus“, „Person“, „Modus“ etc.. Je nach Wortart kann die Selektion der Attribute unterschiedlich sein.

58 Lemmatisierung ist die Invertierung von Deklination oder Konjugation

Der Satz *„Ich fahre mit dem Auto“* wird im ssd-Format folgendermaßen dargestellt:

```
Personalpronomen: {Wort: "ich", Kasus: "Nom.", Person: "1 Pers."
Numerus: "Sing.", Genus : "" },
Verb: {Wort: "fahren", Person: "1. Pers. Sing./Imperativ Sing.",
Tempus: "Praesens", Praefix: "not matched", }
Praeposition: {Wort: "mit", Modus: "Modal,Instrumental,"},
Artikel: {Wort: "dem", Kasus: "Dat./Dat.", Genus: "Mask./Neut."
Numerus: "Sing./Sing." Art: "bestimmt"},
Substantiv: {Wort: "Auto", Genus: "neutrum", Kasus:
"Nom./Akk./Dat." Numerus: "Sing./Sing./Sing.", },
Satzzeichen: {Wort: ".", Zeichen: "Punkt" },
```

Es liegt eine feste Notation vor, die die Elemente mit ihren Attribut-Werte-Paaren beschreibt, und die semistrukturierten Daten haben nun ein ssd-Format angenommen. Die Bezeichnung und die Häufigkeit der Elemente und der Attribute kann beliebig sein. Auch für die Werte gibt es keine Formatvorgaben, wie dies bei strukturierten Daten der Fall ist. Somit bleiben die Sätze der deutschen Sprache durch diese Transformation semistrukturiert; jedoch geraten sie in einen wohlgeformten[59] Zustand.

Ein weiterer Transformationsschritt ist die Phrasen-Analyse, auf die im Kapitel 4 „Phrasen-Analyse“ bereits genauer eingegangen worden ist. Für dieses Kapitel ist nur ein einziger Aspekt relevant: Die Elemente, die eine Aneinanderreihung von Wörtern repräsentieren, sind anderen Elementen, die eine Phrase beschreiben, untergeordnet. In dem Beispielsatz bildet „dem Auto“ eine Nominalphrase, die wiederum einer Präpositionalphrase untergeordnet ist, weil diese Phrase einer Präposition („mit“) folgt.

59 Münz: „Wohlgeformtheit eines XML-Dokuments“, http://de.selfhtml.org/xml/regeln/begriffe.htm#wohlgeformt (2.2007)

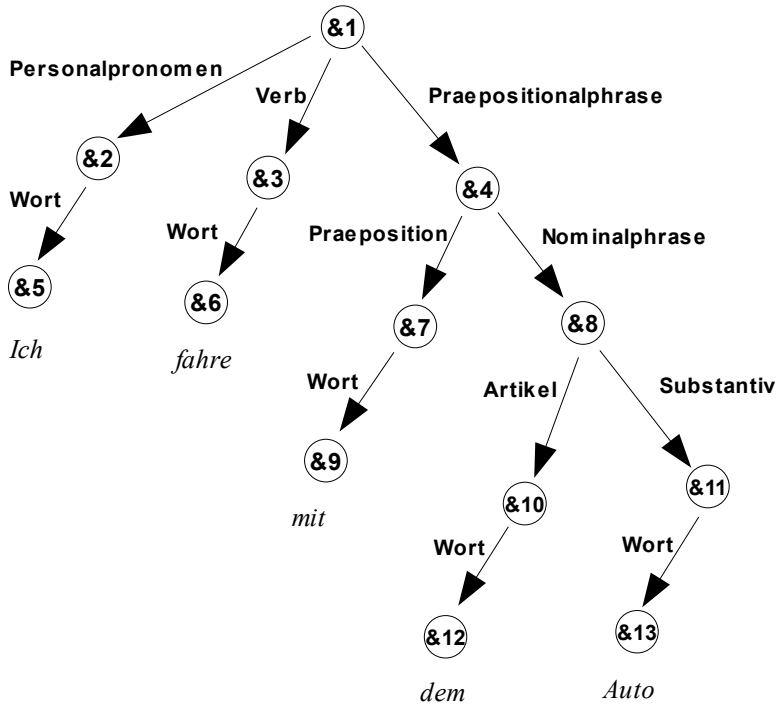


Abbildung 4: OEM-Graph des Satzes "Ich fahre mit dem Auto"

Somit ergibt sich eine Baumstruktur. Innerhalb des OEM-Graphen in Abbildung sind aus Übersichtlichkeitsgründen alle Objekte, außer den atomaren Objekten, die keine Referenz mit dem Attribut Wort haben, eingezeichnet.

Die semistrukturierten Notationen ssd (und auch XML) lassen es zu, dass Elemente in dieser Struktur hierarchisch angeordnet sind:

```
Personalpronomen: {Wort: "ich", Kasus: "Nom.", Person: "1. Pers."
Numerus: "Sing.", Genus : "" },
Verb: {Wort: "fahren", Person: "1. Pers. Sing./Imperativ Sing.",
Tempus: "Praesens", }
Praepositionalphrase: {
    Praeposition: {Wort: "mit", Modus: "Modal,Instrumental,"},
    Nominalphrase: {
         Artikel: {Wort: "dem", Kasus: "Dat./Dat.", Genus:
                          "Mask./Neut." Numerus: "Sing./Sing."
Art:                                    "bestimmt"},
         Substantiv: { Wort: "Auto", Genus: "neutrum", Kasus:
                          "Nom./Akk./Dat." Numerus:
"Sing./Sing./Sing.", }
    },
},
Satzzeichen: {Wort: ".", Zeichen: "Punkt" },
```

5.1 Weitere Strukturierung mit DTDs

In den bisherigen Schritten wurde erreicht, dass semistrukturierte Eingangsdaten so transformiert sind, dass sie in einer wohlgeformten Struktur vorliegen, die es ermöglicht, diese mit der ssd- oder XML-Notation zu beschreiben.

Nun folgt ein weiterer möglicher Schritt, diese transformierten Ausgangsdatensätze zu strukturieren.

Da das Transformationsverfahren und deren enthaltene Algorithmen bekannt sind, ist die Grundmenge aller möglichen Elemente begrenzt und auch bekannt. Ähnliches gilt für die Attribute: Jedes Element hat eine Menge von notwendigen und möglichen Attributen, die ebenfalls bei den Erkennungsverfahren der einzelnen Wortgruppen bestimmt werden.

Wissen wir, welche Wortarten (Elemente) es gibt, so können wir über einen oder mehrere mit dem gleichen Verfahren transformierten Sätze der deutschen Sprache eine für alle gültige DTD (siehe Kapitel 2.4.3 „Document Type Definitions DTDs“) erstellen.

5.2 Alternative Wege für strukturierte Datenhaltung

Innerhalb der ER-Modellierung (siehe strukturierte Datenhaltung) bietet sich die Möglichkeit an, für jedes Wort-Element oder Satz-Element eine eigene Entity anzulegen. Da jedoch in Sätzen der Natursprache auch die Reihenfolge der Elemente für die weiterführende, semantisch orientierte Verarbeitung von Relevanz ist, ist es unerlässlich, dass diese Elemente aneinander gekettet werden müssen.
Etwas Schwierigkeiten bei diesem Ansatz macht die Tatsache, dass bei Referenzen (Constraints) die Entities genau festgelegt werden müssen. Somit ist es nicht auf direktem Wege möglich, von einem beliebigen wortbeschreibenden Entity zu einem anderen beliebigen zu referenzieren.

Eine andere Möglichkeit bietet die Spezifikation einer einzigen Entity, mit der alle Wortarten beschrieben werden können. Das Entity enthält das Wort selber, evtl. die Grundform und weitere Attribut-Werte-Paare, die das Wort näher beschreiben.
Dies entspricht der Vereinigungsmenge aller Attribute aller wortbeschreibenden Elemente der semistrukturierten Datenhaltung (s. u.). Dieses Verfahren ist prinzipiell sehr gut anwendbar, da die Vereinigungsmenge aller Attribute überschaubar ist. In den Diplomarbeiten von Dischert/Kiefel und Johann wurde dieses Verfahren angewandt.

Fazit:

Dieses Kapital hat gezeigt, wie die Konzeption der semistrukturierten Daten für unser Analysesystem angewandt wird. Dabei ist die Verarbeitung und Beschreibung der semistrukturierten Sätze drei Stufen durchlaufen, die dazu führen, dass wir uns der strukturierten Datenhaltung immer weiter annähern, jedoch definitionsgemäß diese nicht erreichen.

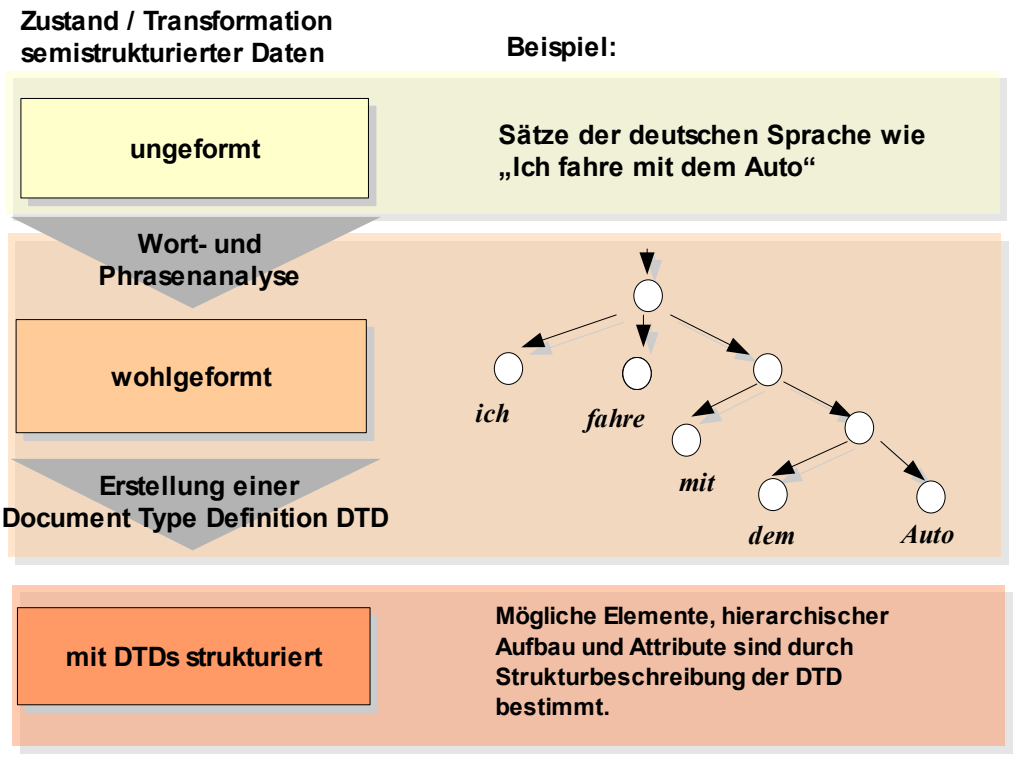


Abbildung 5: Zustand und Transformation semistrukturierter Daten

6 Realisierung

Nachdem das Analysesystem auf seiner funktionalen Ebene beschrieben ist, widmen sich die folgenden Kapitel der Realisierung auf technischer Ebene. Wie schon in der Einleitung erwähnt, stützt sich die Wort- und Phrasen-Analyse vollständig auf das von mir entwickelte Software-System IPEE (Internet Pattern Extract Engine), dessen interne Struktur, besonders die Komponenten, die in dieser Masterarbeit Anwendung finden, in den folgenden Kapiteln 7 bis 9 beschrieben werden.

In diesem Kapitel wird kurz auf die Beweggründe der Entwicklung von IPEE eingegangen und auf die gestellten Projektanforderungen. Es wird dargelegt, wie die Anforderungen für die praktischen Teile dieser Masterarbeit mit IPEE abgedeckt werden können.
In den vergangenen Kapiteln wurden allgemeine Verfahren zur semistrukturierten Datenhaltung und Abfrage diskutiert und auch deren Einschränkungen dargestellt.
In diesem Kapitel soll beleuchtet werden, dass die Abfrage von Informationen aus Fließtexten oder HTML-Dokumenten mit den beschriebenen, allgemeinen semistrukturierten Abfragesprachen ebenfalls eingeschränkt ist, jedoch mit der Konzeption von IPEE realisierbar.

6.1 Projektanforderungen von IPEE

Zur allgemeinen Beschreibung der grundlegenden Projektanforderungen, die sich aus dem Gespräch mit dem Kunden ergaben aber auch aus selbst gesteckten Zielen, soll aus meinem Praxis-Semester-Bericht zitiert werden:“

- Die Beschreibung der Patterns und Datensätze soll flexibel für verschiedene Bedürfnisse eingesetzt werden können,
- einfach wartbar (anpassbar),
- übersichtlich und leicht zu erlernen sein.
- Unter-Patterns, die Formate beschreiben (wie verschiedene Datumsformate oder numerische Formate), sollen wiederverwendbar sein.
- Das Matchingverfahren soll sowohl Tabellenstrukturen als auch fließende Texte verarbeiten können; für beides sollen die gleichen Patterns verwendet werden.
- Das Grundkonzept der Softwarelösung soll auch für andere Zwecke einsetzbar und leicht anpassbar sein. In Planung ist, dieses Produkt als Open-Source-Software (und evtl. für weitere Kunden Support-, Wartungs- und Customizing-Leistungen) anzubieten.“ [60]

60 Frey: „ICEE – Internet Car Extract Engine“ S. 4

Aus diesen Anforderungen wurde von mir die Konzeption von IPEE als Software-Anwendung entwickelt. Dieses Kapitel und die folgenden sollen zeigen, wie sich die Realisierung auch für die Wort- und Phrasen-Analyse und die Transformationen eines Satzes der deutschen Sprache in eine wohlgeformte, semistrukturierte Darstellung eignet.

6.2 Einsatz von IPEE als Meta-Suchmaschine

Wie schon in der Einleitung erwähnt, war der ursprüngliche Beweggrund zur Entwicklung von IPEE die Realisierung einer Meta-Suchmaschine für verschiedene Automobilhandelsbörsen.
Mit Hilfe eines Crawling-Verfahrens und mit der Pattern-Extraktion sollen Fahrzeuginformationen in eine einheitliche Struktur transformiert werden, so dass die resultierenden Fahrzeugdaten, die aus unterschiedlichen Quellen stammen, mittels einer DTD (siehe Kapitel 2.4.3 „Document Type Definitions DTDs“) strukturierbar sind.

Für das Crawling-Verfahren enthält IPEE eine eigene Komponente, die als Crawler bezeichnet wird, welche in der Lage ist, aus den vom Anwender ausgehenden Fahrzeugdaten eine Suchanfrage speziell für eine bestimmte Automobilbörse im Internet zu generieren. Diese Fahrzeugdaten bestehen aus unterschiedlichen Parametern, wie „Hersteller“, „KMStand von/bis“, „Erstzulassung von/bis“ „KW/PS von/bis“ etc. Der Crawler generiert anhand der vom Webservice abgefragten HTML-Seite zur Eingabe dieser Parameter eine URL, die die Trefferliste erwünschter Automobile referenziert.

Innerhalb der Wort- und Phrasen-Analyse werden zwar ebenfalls Suchanfragen auf die Internet-Lexika wissen.de und canoonet gestellt. Das Suchwort wird lediglich durch Zusammensetzen von Strings in eine URL umgewandelt. Deswegen wird diese Komponente in dieser Masterarbeit nicht weiter dokumentiert.

Weiterführend können mit der Anwendung von Patterns und einer nachfolgenden Transformation unterschiedliche Datumsformate erkannt und in eine einheitliche Struktur übersetzt werden, so dass die resultierenden Daten sich noch weiter der Form von strukturierten Daten annähern. Diese aus unterschiedlichen Quellen stammenden Daten können nun mit einer strukturierten RDBMS abgelegt und mit OLAP bzw. dem Datawarehousing-Verfahren[61] ausgewertet werden.

61 Verfahren zur graphischen Auswertung von (relationalen) Datenbanken

6.3 Die Anwendung IPEE für den Einsatz von Wort- und Phrasen-Analyse

Die Anforderung, Fließtexte mit Hilfe von Patterns zu extrahieren und Zeichenkettenfolgen zu erkennen, die in einer bestimmten Konstellation oder Reihenfolge auftreten, findet auch innerhalb der Wort- und Phrasen-Analyse ihren Einsatz. Aus diesem Grund wurde der Gedanke verfolgt, diesen Aufgabenbereich mit Hilfe der Konzeption von IPEE zu unterstützen.
Bei flektierten Wörtern kann mit Hilfe von Patterns die Endung eines Wortes erkannt werden, und der Stamm wird als extrahiertes Element ausgegeben. Auch Phrasen wie Nominal- und Präpositionalphrasen werden mit Hilfe eines Patterns erkannt. Diese Patterns erkennen mögliche Konstellationen von Wortarten, die bei einem Matching (Erkennung) ausgegeben werden. Auf dieses Verfahren wird in Kapitel 10 „Implementiertechnische Aspekte der Wort- und Phrasen-Analyse“ noch genauer eingegangen.

Wie bei der Anwendung als Meta-Suchmaschine werden Patterns eingesetzt, um die benötigten Informationen der Wörterbücher zu extrahieren. Im folgenden wird IPEE mit seinen Komponenten, die innerhalb der Wort- und Phrasen-Analyse eingesetzt werden, erklärt.

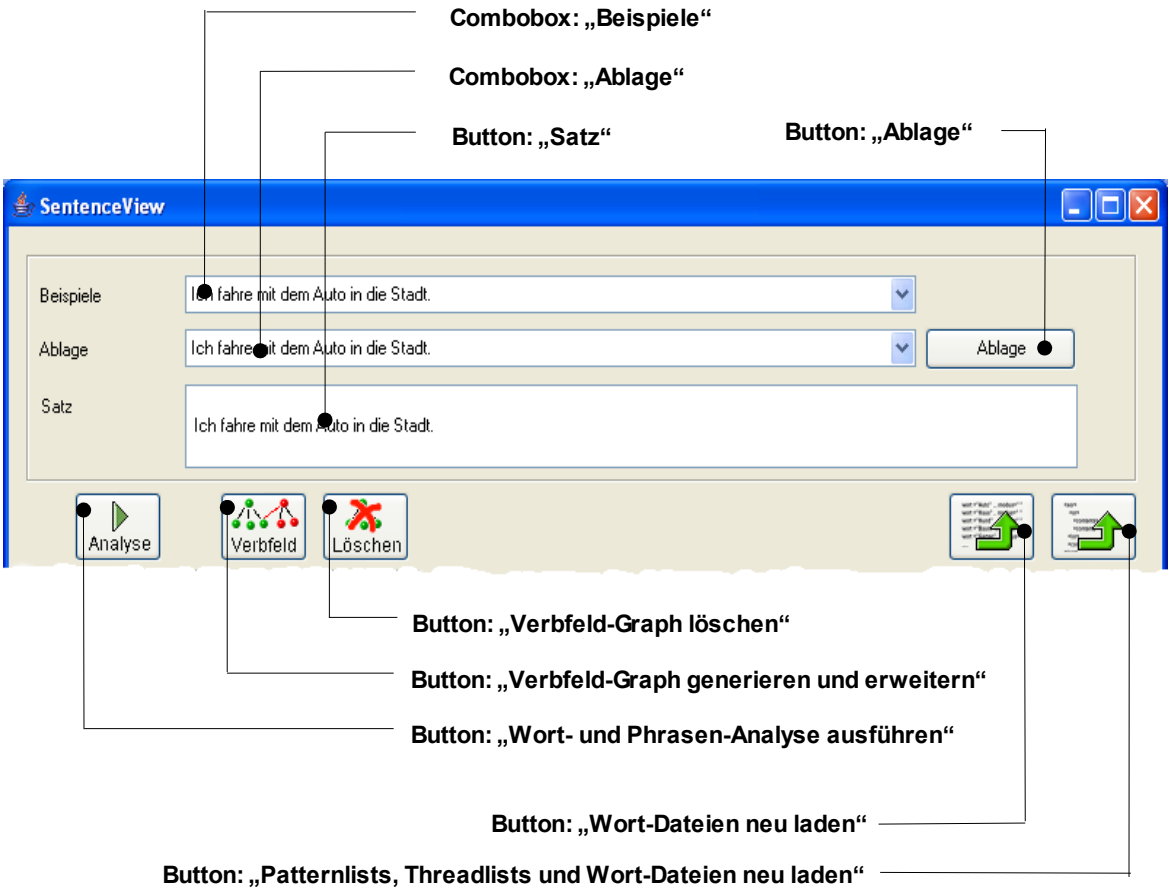


Abbildung 6: Ausschnitt aus der Oberfläche von SentenceView

6.4 Die Anwendung SentenceView

Für die experimentellen Arbeiten der Wort- und Phrasen-Analyse und für die semantisch orientierte Verarbeitung von Verben (siehe Kapitel 11 „Semantisch orientierte Verarbeitung“) wurde innerhalb dieser Masterthesis die graphische Benutzeroberfläche SentenceView entwickelt.
Die Oberfläche SentenceView enthält ein Textfeld[62] (siehe Abbildung 6) mit der Beschriftung „Satz“ (im folgenden Eingabe-Textfeld genannt), in das vom Anwender ein Satz oder Wörter der deutschen Sprache eingegeben werden können.
Zusätzlich existiert innerhalb der Oberfläche eine Combobox „Beispiele“[63], innerhalb der bestimmte vorgegebene Beispielsätze ausgewählt werden können und eine weitere Combobox „Ablage“, die alle Eingaben innerhalb des Eingabe-Textfeldes bei Anklicken der Buttons „Verbfeld-Graph generieren und erweitern“, „Wort-und Phasen-Analyse ausführen“ oder mit dem Button „Ablage“ zwischenspeichert.

62 graphisches Element zur Eingabe eines Textes
63 graphisches Element zur Auswahl von Zeichenketten

Durch Anklicken der Buttons[64] (siehe Abbildung 6) können Anwendungsfälle ausgelöst werden, die im folgenden Unterkapitel beschrieben sind.
Die Ausgabe (Ergebnis der Anwendungsfälle) erfolgt auf graphischen Kontroll-Elementen für die Ansichten von Tabellen (im folgenden als graphische Tabellen-Elemente bezeichnet). Diese graphischen Tabellen-Elemente können durch Anklicken der Tabwriter[65] sichtbar gemacht werden

graphische Tabellen-Elemente, die durch Tabwriter sichtbar zu machen sind und folgende Beschriftung haben	**Inhalt der graphischen Tabellen-Elemente**
Wort-Analyse	Ausgabe der Wörter mit ihren erkannten Attributen
Phrasen-Analyse	Ausgabe der Satz-Phrasen, die den hierarchischen Aufbau (Verschachtelung der Phrasen in einer ebenen Struktur) wiedergibt
Verb aus wissen.de	Angabe der Lesarten des Verbs, die aus dem Internet-Lexikon wissen.de recherchiert sind. Eine Lesart enthält grammatische Angaben, Bedeutungsnummer, Bedeutungserklärung + weitere Angaben zu Bedeutungserklärungen (diese sind unterhalb der jeweiligen Bedeutungserklärung angegeben).

Tabelle 5: Tabwriter innerhalb SentenceView

6.5 Klassen innerhalb der Anwendung SentenceView

Die Klasse `SentenceView` ist die Main-Klasse dieser Anwendung und instanziert beim Aufruf eine Klasse `SentenceViewFrame,` die von der Klasse `JFrame` von der Java.Swing Api[66] abgeleitet ist und ein Dialogfenster auf dem Bildschirm repräsentiert.

Die Klasse `SentenceViewFrame` ist Klassen übergeordnet, die direkt nach dem Programmstart folgende Klassen (jeweils einmal) instanziert - siehe auch Abbildung 7.

- Die Klasse `IPEEAccess` dient zur Durchführung und Nachverarbeitung von Zugriffen auf die Softwarelösung IPEE, die hier zur Durchführung der Wort- und Phrasen-Analyse eingesetzt wird.

- Klasse `VerbFieldGraphGenerator` für die Generierung des Verbfeld-

64 graphisches Element, welches bei Anklicken mit dem Mauspfeil eine Methode innerhalb eines Programms ausführt
65 graphisches Element, um bestimmte untergeordnete Oberflächen sichtbar zu machen
66 Klassenbibliothek zur Implementierung graphischer Benutzeroberflächen für die Programmiersprache Java

Graphen (siehe Kapitel 11 „Semantisch orientierte Verarbeitung“)

- Klasse `BedWBManger` für die Verwaltung eines Bedeutungswörterbuches mit Hilfe einer XML-Datei. (siehe Unterkapitel 6.6 „Bedeutungs-Lexikon“)

- Klasse `ssdConverter` für die Konvertierung von Instanzen des Typs `ssdNode` in Zeichenketten, die der Notation ssd entsprechen und umgekehrt

Bei Instanzierung der Klasse `SentenceViewFrame` wird die Klasse Methode `reloadPatternlistAndThreadlist()` von `IPEEAccess` aufgerufen. Diese instanziert nachfolgende Klassen, die zur Klassenbibliothek von IPEE gehören und im nächsten Unterkapitel zu finden sind.

Dazu gehören die Klassen `ThreadObjects` und `ThreadList`. Die Klasse `ThreadList` dient dazu, sog. Threadlist-XML-Datei innerhalb IPEE auszuführen. Threadlist-XML-Datei beschreiben eine Anweisungsliste.

Jedes Element innerhalb einer Threadlist-XML-Datei beschreibt eine bestimmte Klasse, die ausgeführt und der anhand ihrer Attribute-Werte-Tupels bestimmte Parameter übergeben werden sollen. Diese Klassen enthalten eine Methode `run()`, die wiederum bestimmte Variablen (siehe Kapitel 8 „Design und implementiertechnische Aspekte der Threadlists“) aus der Instanz der Klasse `ThreadObjects` ausliest, einen bestimmten verarbeitenden Code-Block ausführt und das Ergebnis der Verarbeitung als sog. Threadlist-Variable wieder zurück in die Instanz `threadObjects` schreibt.

Eine genaue Beschreibung der Ausführung von Threadlist-XML-Datei folgt im Kapitel 8 „Design und implementiertechnische Aspekte der Threadlists“.

Die Aufgaben der verarbeitenden Code-Blöcke lassen sich semantisch wie folgt beschreiben:

- Laden / Speichern einer Datei
- Laden einer HTML-Seite
- Vorbereitungen von Patterns
- Ausführung von Pattern-Extraktionsverfahren
- Anlegen / Zusammensetzen / und Kopieren von Strings
- Ausführen von Kontrollstrukturen wie If/Else-Blöcke / While-Schleifen
- Aufrufen von untergeordneten Threadlist-XML-Datei

Die Klasse `ThreadObjekts` verwaltet sämtliche Instanzen, die für die Ausführung der Elemente, die von `AbstractThreadElement` abgeleitet sind, verwendet werden. Dazu gehören auch die Threadlist-Variablen.

Es werden folgende Instanzen verwendet innerhalb der Methode

`reloadPatternListAndThreadList():`

Die Instanz `prepareWordAnalyseThreadList` von der Klasse `ThreadList` für Vorbereitung der Wort-Analyse

Die Instanz `wordAnalyseThreadList` von der Klasse `ThreadList` für Vorbereitung der Wort-Analyse

Die Instanz `phraseAnalyseThreadList` von der Klasse `ThreadList` für Durchführung der Phrasen-Analyse

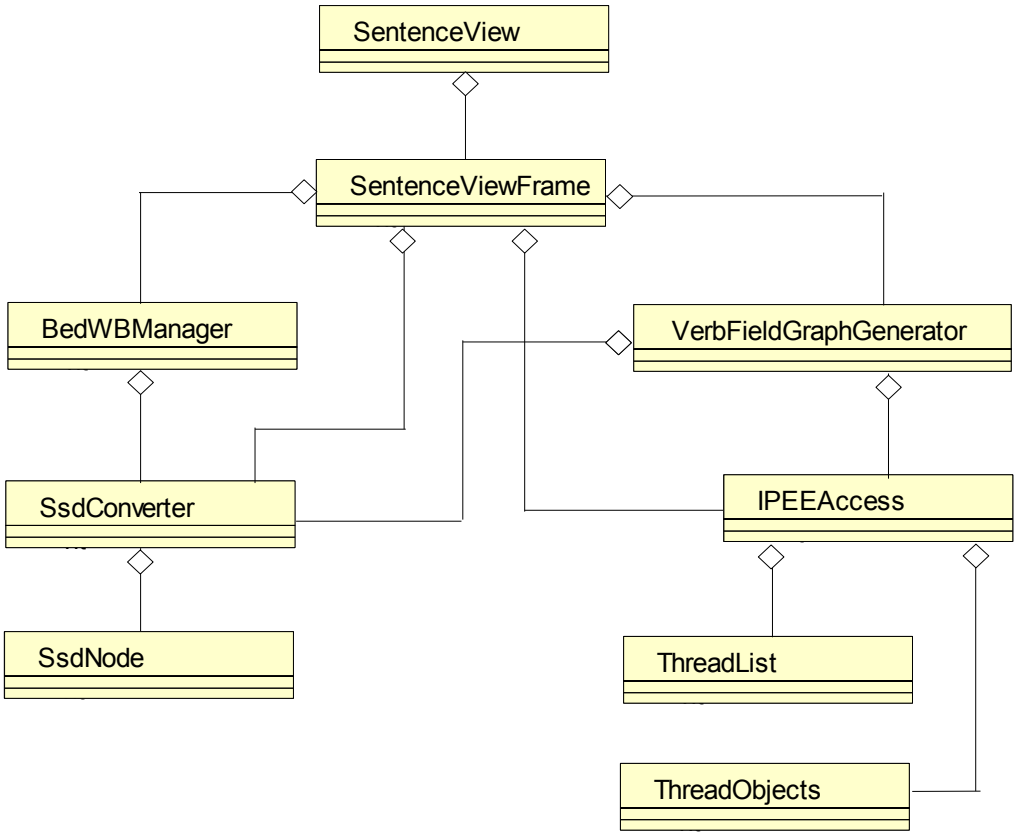


Abbildung 7: UML-Klassendiagramm von SentenceView mit Zugriffen auf IPEE

Die Instanz der Klasse `ThreadObjekts;` diese wird allen `ThreadList` Instanzen übergeben.

6.6 Bedeutungs-Lexikon

Während der Anwendung von SentenceView wird für die Verben zusätzlich ein

Bedeutungslexikon gepflegt. Das Bedeutungslexikon ist eine XML-Datei `BedWB/BedWB.xml`, die mit der DTD `BedWB/WebWB.dtd` strukturiert ist.

Diese Datei wird mit dem direkten Programmaufruf der Methode `loadBedWBfromFile()` von der Klasse `BedWBManager` erzeugt und enthält Elemente für Wörterbuchartikel (`WArtikelElemente`), die Verben als Schlagwort enthalten (`SchlagWElement`). Zusätzlich enthält jedes `WArtikelElement` einen Artikeleintrag (`ArtEinElement`). Dieses `ArtEinElement` hat wiederum weitere Elemente, darunter ein `BErklElement` für Bedeutungserklärungen und eine `BErWAElement` für weitere Angaben zur Bedeutungserklärung, die Beispielanwendungen des Verbs wiedergeben.

Für die Erstellung des Bedeutungswörterbuchs sind die Klassen `BedWBManager`, `BedWBElement` und `BedWBElementGenerator` zuständig.
Die Klasse `BedWBManager` enthält eine Methode `main()`, die die `BedWB.xml` Datei neu generiert. `Die BedWB.xml` wird beim Aufruf der Methode `main()` nach dem oben beschriebenen Muster mit den Verben gefüllt, die in der `Nebsy/Verb.DAT` Datei eingetragen sind (179 Verben), und der `BedWB/VerbenAddOn.ssd` Datei (134 Verben), die von Professor Dr. phil. Gregor Büchel zusätzlich erstellt worden ist[67] (siehe Anhang C. 4).

Die Klasse `BedWBManager` wird auch von SentenceView instanziert. Es wird für alle Verben, die innerhalb SentenceView verarbeitet werden, die Methode `extendBedWB(String SchlagW)` von `BedWBManager` aufgerufen. Dies kann innerhalb der entsprechenden verarbeitenden Methoden bei der Bearbeitung von Anwendungfällen wie „Verbfeld-Graph generieren“ oder „Wort- und Phrasen-Analyse ausführen“ (siehe Kapitel 6.7. „Anwendungsfälle innerhalb SentenceView“) durchgeführt werden.

Diese Methode prüft, ob das Verb in `BedWB` schon vorhanden ist. Sollte dies nicht der Fall sein, wird die `BedWB.xml` entsprechend erweitert.

6.7 Anwendungsfälle innerhalb SentenceView

Der Benutzer hat die Möglichkeit, folgende Anwendungsfälle durchzuführen, die jeweils durch Buttons mit der entsprechenden Bezeichnung ausgelöst werden können:

1. Wort- und Phrasen-Analyse ausführen
2. Verbfeld-Graph generieren und erweitern
3. Verbfeld-Graph löschen
4. Wort-Dateien neu laden
5. Patternlists, Threadlists und Wort-Dateien neu laden

67 Büchel: [Zusätzliche Verben für das Bedeutungswörterbuch], siehe Anhang C.4

Im folgenden sollen die wesentlichen Anwendungsfälle „Wort- und Phrasen-Analyse ausführen“ und „Verbfeld-Graph generieren“ beschrieben werden.

6.7.1 Der Anwendungsfall „Wort- und Phrasen-Analyse ausführen“

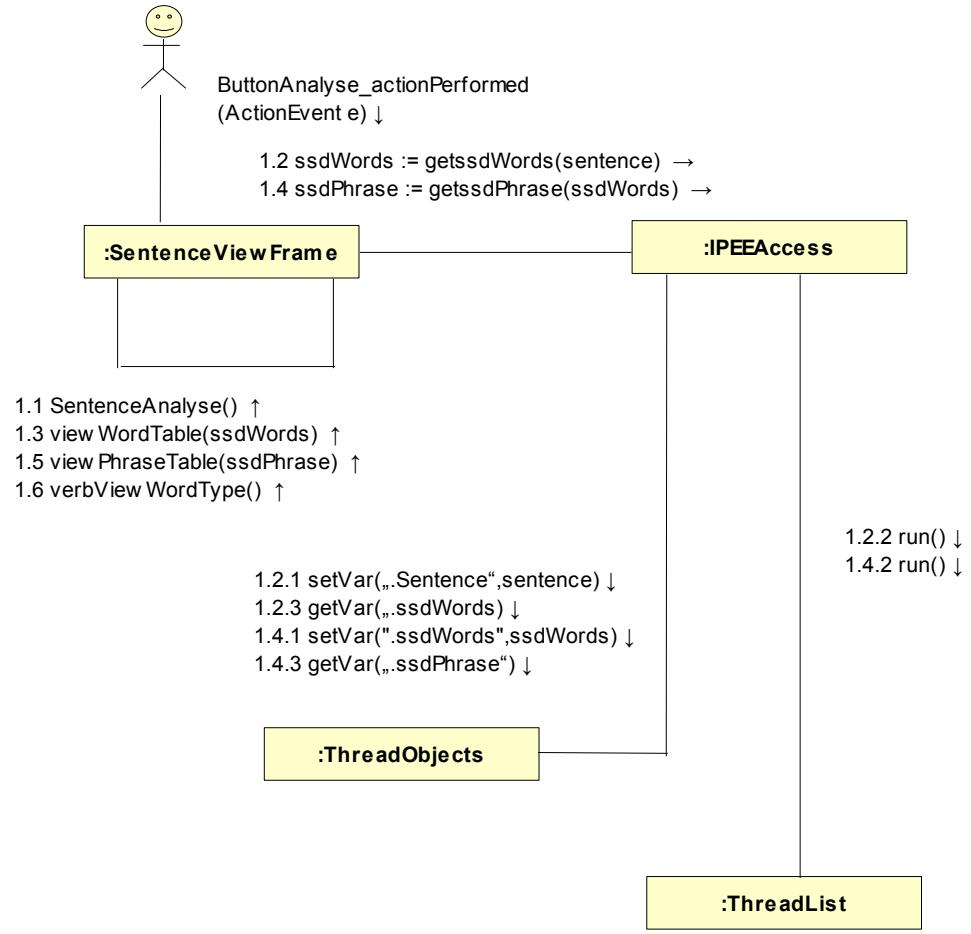


Abbildung 8: UML-Kollaborationsdiagramm für Anwendungsfall "Wort- und Phrasen-Analyse ausführen"

Bei Anklicken des Buttons: „Wort- und Phrasen-Analyse ausführen„ wird die Methode

`ButtonAnalyse_actionPerformed(ActionEvent e)` der Klasse `SentenceViewFrame` aufgerufen. Diese Methode ruft die Methode `SentenceAnalyse()` innerhalb der gleichen Klasse auf.
Darauf wird der eingegebene Satz aus der Instanz `jTextFieldSentence` der Klasse `Jtextfield`[68], die das Eingabefeld für den vom Benutzer eingegebenen Satz repräsentiert, ausgelesen. (Diese Klasse ist nicht im Kollaborationsdiagramm siehe Abbildung 8 eingezeichnet, da graphische Kontrollelemente nicht berücksichtigt werden sollen.)
`sentence = jTextFieldSentence.getText();`
Der eingegebene Satz wird in die Stringvariable `sentence` geschrieben.

Die Methode `String ssdWordType =getssdWordType(sentence)` von der Klasse `IPEEAccees` führt Methodenaufrufe auf Instanzen von Klassen durch, die der Klassenbibliothek von IPEE angehören.

Die Methode `setVar(„.Sentence",sentence)` schreibt den eingegebenen Satz (`sentence`) als Threadlist-Variable `„.Sentence"` in eine Instanz der Klasse `ThreadObjects`, die zuvor angelegt und den Instanzen der Klassen von `ThreadList` übergeben wurde.

Danach wird die Instanz `wordAnalyseThreadList` der Klassen `ThreadList` mit `run()` aufgerufen.

Diese wurde zuvor mit folgendem Konstruktoraufruf innerhalb der Methode `reloadPatternlistAndThreadlist()` von `IPEEAccess` instanziert:
```
wordAnalyseThreadList = new
ThreadList("SentenceTossd/WordAnalyse/WordAnalyse.xml");
```

Dabei wird der Pfad der entsprechenden Threadlist-XML-Datei `SentenceTossd/WordAnalyse/WordAnalyse.xml` übergeben, die bei dem Aufruf der Methode `run()` ausgeführt wird. Durch die Ausführung dieser Datei wird die Wort-Analyse durchgeführt (siehe Kapitel 10 „Implementiertechnische Aspekte der Wort- und Phrasen-Analyse").

Die Methode `getVar(„.ssdWords")` liest den wortanalysierten Satz, der nun in ssd-Notation vorliegt, aus. Dies ist auch der Rückgabewert der Methode `getssdWords()` und wird in Klasse `SentenceViewFrame` in die Variable `ssdWords` übergeben.

Die Methode `viewWordTable(ssdWords)` schreibt das Ergebnis der Wort-Analyse in das graphische Tabellen-Element, das mit Anklicken des Tabwriters „Wort-Analyse" aktivierbar ist.

Die Methode `ssdPhrase =getssdPhrase(ssdWords)` der Klasse `IPEEAccess`

68 Klasse aus der Klassenbibleothek java.swing für grafische Kontrollelemente zur Texteingabe

führt auf Basis des wortanalysierten Satzes die Phrasen-Analyse durch (siehe dazu Kapitel 4 „Phrasen-Analyse“ und Unterkapitel 10.5 „Erkennung von Nominalphrasen“).

Die Verarbeitung innerhalb der Methode `getssdPhrase(ssdWords)` läuft synonym zu der Verarbeitung der Methode `getssdWordType(sentence)` ab. Deswegen soll diese verkürzt erklärt werden:

Es folgt ein Aufruf mit der Methode `run()` auf die Instanz `phrasenAnaylseThreadList,` die zuvor mit der Threadlist-XML-Datei „`SentenceTossd/PhraseAnalyse/PhraseAnalyse.xml`“ instanziert wurde. Diese Instanz führt aus dem Satz, der in die `threadObjects` Instanz übergeben wurde, die Phrasen-Analyse durch.

Die Methode `viewWordTable(ssdWords)` schreibt das Ergebnis der Phrasen-Analyse in das graphische Tabellen-Element der Oberfläche, die mit Anklicken des Tabwriters „Phrasen-Analyse“ aktivierbar ist.

Die Methode `verbViewWordTable()` sucht sich das erste Verb aus dem eingegebenen Satz heraus, führt einen Zugriff auf das Internet-Lexikon wissen.de und zeigt die Lesarten (siehe Kapitel 11.1 „Zuordnung eines Hyperonyms zum Verb durch Recherche aus dem Wörterbuch [wissen.de]“) des Verbs in dem graphischen Tabellen-Element der Oberfläche, die mit Anklicken des Tabwriters „Verb aus wissen.de“ aktivierbar ist.

6.7.2 Der Anwendungsfall „Verbfeld-Graph generieren“

Mit Anklicken des Buttons „Verbfeld-Graph generieren und erweitern“ wird die Methode `jButtonVerbTree_actionPerformed` ausgelöst; innerhalb SentenceViewFrame wird mit der eigenen Methode `sentenceAnalyse()` wie im vorigen Anwendungsfall eine Wort- und Phrasen-Analyse ausgeführt.

Darauf wird mit Methode `appendSentenceToVerbFieldGraph(ssdPhrase,` von der Klasse `VerbGraphGenerator` aufgerufen, die Generierung oder Erweiterung des Verbfeld-Graphen durchführt. Die Generierung des Verbfeld-Graphen ist in Kapitel 11.3 „Realisierung des semantisch In-Beziehung-Setzens von Verben“ beschrieben.

7 Element-Transformation

Dieses Kapitel ist aus Gründen des Copyrights, die auf der Software IPEE liegen, nicht in dieser veröffentlichten Masterarbeit enthalten (siehe „Änderungen und Einschränkungen der KOPS-Veröffentlichung aus Gründen des Copyrights“)

8 Design und implementiertechnische Aspekte der Threadlists

Dieses Kapitel ist aus Gründen des Copyrights, die auf der Software IPEE liegen, nur ausschnittweise in der veröffentlichten Masterarbeit enthalten, um bestimmte Begriffe zu klären wie Threadlists, Threadlist-XML-Datei, Threadlist-Variablen, da das Kapitel 10 darauf aufbaut.

Threadlists dienen (angelehnt an sog. 4GL[69]) zur Programmierung der Pattern-Extraktion von IPEE und deren weiteren Verarbeitung.

Eine Threadlist enthält auch bestimmte Elemente zur Verarbeitung einer Kontrollstruktur wie „`if`“, „`else`“, „`while`“, „`include`“ und „`call`“.

Eine Threadlist stellt eine Anweisungsliste dar. Diese Threadlists werden mit XML-Dokumenten beschrieben, in denen die Anweisungen nacheinander aufgelistet und mit einem `ThreadlistElement` (ein XML-Tag mit der Bezeichnung `Threadlist`) eingehüllt sind. (Diese XML-Dokumente werden Threadlist-XML-Datei genannt.)

Die Werte können entweder konstante Zeichenketten sein oder Variablen, die im folgenden „Threadlist-Variablen“ genannt werden und folgende Notation haben: Grundsätzlich werden Variablen mit einem Objektnamen (Präfix) beschrieben; dieser kann ein oder mehrere Variablennamen (Suffixe) haben.

Eine Threadlist-Variable kann nur mit ihrem Objektnamen angegeben sein und wird folgendermaßen notiert:

```
Attribute=".Objektname"
```

Eine Threadlist-Variable kann somit mehrere Variablennamen haben:

```
Attribute="Objektname.Variablename1"
Attribute="Objektname.Variablename2"
Attribute="Objektname.Variablename3"
```

Im folgenden wird eine vereinfachte Threadlist-XML-Datei beschrieben, die dazu dient, regelmäßige Verben zu erkennen.

69 Programmiersprachen der vierten Generation (Fourth generation language). Diese sind darauf ausgerichtet, rasch für einen bestimmten Anwendungsbereich Prozeduren schreiben zu können.

Gegenüber der Verb-Erkennung des realisierten Prototyps dieser Masterarbeit, werden hier folgende Aspekte nicht berücksichtigt:

- Erkennung von unregelmäßigen Verben
- Erkennung von Präfixen
- Existenz-Überprüfung der Verben an Hand automatischer Recherche aus dem Internet-Lexikon wissen.de
- Erkennung von Verben, deren Infinitiv mit dem Stamm und der Endung „n“ gebildet wird, wie "nähern", "sammeln", "erinnern" ...
- Erkennung von Perfekt-Formen

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE threadlist SYSTEM "ipee\dtd\threadlist.dtd">
<threadlist>

    ...
    <wordextractor patternlist="VerbPattern" in=".Word"
          out=".Result"/>
     <if operation="notequal" operant1="Result.Infinitiv"
               operant2="not matched">
          <concat in="en" to="Result.Infinitiv"/>

          <include
               file="SentenceTossd/WordAnalyse/Concat/Verb.xml"/>

    </if>
    ...
</threadlist>
...
```

Diese Threadlist-XML-Datei ist jedoch nicht unabhängig, sondern wird von einer übergeordneten Threadlist aufgerufen. Die aufrufende Threadlist muss folgende Vorbereitungen treffen:

- Die Patternlist `VerbPattern` muss vorbereitet sein.
- In der Threadlist-Variable `.Word` steht ein zu analysierendes Verb wie „ lerne “.

Das Threadlist-Element

```
    <wordextractor patternlist="VerbPattern" in=".Word"
          out=".Result"/>
```

veranlasst den Aufruf des `WordExtraktors` , der mit Hilfe der Patternlist `VerbPattern` den Inhalt der Threadlist-Variable `.Word` extrahiert.

Der Wordextraktor ist eine speziell ausgelegte Variante des Extraktors (siehe Kapitel 6.3 “Die Anwendung IPEE für den Einsatz von Wort- und Phrasen-Analyse“). Eine genauere Beschreibung von Patternlists und der Extraktion soll aus Gründen des Copyrights, die auf der Software IPEE liegen, nicht in die veröffentlichte Masterarbeit mit einfließen.

Das Ergebnis dieser Extraktion wird in `.Result` ausgegeben:

Threadlist-Variable		**Funktion**
Objektname	**Variablenname**	
`Result.`	`Infinitiv`	Stamm, aus dem die Grundform gebildet wird
`Result.`	`Person`	Person des Verbs
`Result.`	`Zeit`	Zeit des Verbs

Bei Fehlschlagen der Extraktion wird in den ausgehenden Threadlist-Variablen die Zeichenkette „not matched“ eingetragen.

Wenn das Verb mit Hilfe des Extraktionsverfahrens analysiert werden kann, gilt folgende if-Anweisung als erfüllt, weil in der Threadlist-Variable `Result.Infinitv` ein gefundener Wert steht und nicht „not matched“.

```
<if operation="notequal" operant1="Result.Infinitiv"
        operant2="not matched">
```

Das folgende Element hängt die Zeichen „en“ an den extrahierten Stamm des Verbs in der Threadlist-Variablen `Result.Infinitiv`.

```
<concat in="en" to="Result.Infinitiv"/>
```

Das folgende Element ruft eine weitere Threadlist-XML-Datei `ConcatVerb.xml` auf. Die Elemente dieser Threadlist-Datei werden so ausgeführt, als ob sie in den aufrufenden Threadlists stehen.

```
<include file="ConcatVerb.xml"/>
```

Diese XML-Datei `ConcatVerb.xml` fügt die Attribute der Threadlist-XML-Datei zu einem String zusammen, dessen Inhalt dem ssd-Format entspricht.

Die Threadlist-XML-Datei `ConcatVerb.xml` sieht folgendermaßen aus:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE threadlist SYSTEM "ipee\dtd\threadlist.dtd">
<threadlist>
    <concat in="Verb: {Wort: &quot;" to=".ssdWord"/>
    <concat in="Result.Infinitiv" to=".ssdWord"/>
    <concat in="&quot;, Person: &quot;" to=".ssdWord"/>
    <concat in="Result.Person" to=".ssdWord"/>
    <concat in="&quot;, Tempus: &quot;" to=".ssdWord"/>
    <concat in="Result.Zeit" to=".ssdWord"/>
    <concat in="&quot;, " to=".ssdWord"/>
    <concat in="}" to=".ssdWord"/>
</threadlist>
```

Ergebnis:

Bei Übergabe der Zeichenkette „lerne“ in `.Word` und Ausführung der Threadlist zur Erkennung regelmäßiger Verben steht folgendes Ergebnis in `.ssdWord`:

```
Verb: {Wort: "lernen", Person: "1. Pers. Sing./Imperativ Sing.",
Tempus: "Praesens" }
```

9 Design und implementiertechnische Aspekte der Extraktion

Dieses Kapitel ist aus Gründen des Copyrights, die auf der Software IPEE liegen, nicht in der veröffentlichten Masterarbeit enthalten (siehe „Änderungen und Einschränkungen der KOPS-Veröffentlichung aus Gründen des Copyrights“).

10 Implementiertechnische Aspekte der Wort- und Phrasen-Analyse

Nachdem nun in den vergangenen Kapiteln IPEE und die Verarbeitung von Threadlists und Patternlists beschrieben wurde, soll der Fokus in diesem Kapitel auf wesentliche, spezielle Aspekte des Designs der Wort- und Phrasen-Analyse gerichtet sein.
Um die Ausführungen im Rahmen zu halten, ist die Beschränkung auf das Wesentliche wichtig und durchaus auch angebracht, weil die Komponenten der Abläufe für die Erkennung von Wörtern sehr starke strukturelle Ähnlichkeiten aufweisen.

10.1 Einladen der Wort-Dateien

Für die Vorbereitung der Wort- und Phrasen-Analyse eines Satzes wird
IPEE mit der Threadlist
`´SentenceTossd/PrepareDate/PrepareWordAnalyse.xml` aufgerufen.

Dies geschieht mit der Methode `run()`, der von SentenceView angelegten Instanz `prepareWordAnaylseThreadList` von der Klasse `ThreadList`.

Diese Threadlist sorgt für Initialisierung aller benötigten Patterns und lädt die *.ssd (aus dem Verzeichnis `WordData/`) und *.DAT-Dateien (aus dem Verzeichnis `Nebsy/`) (siehe Kapitel 3.1 „Einfache Wörter“).

Die Dateien werden in Threadlist-Variablen (siehe Kapitel 8 „Design und implementiertechnische Aspekte der Threadlists“) eingeladen und sind folgendermaßen wiederzufinden:

Threadlist-Variable Variablenname	Beschreibung des Inhaltes	wird generiert durch die Threadlist-XML-Datei	zugrunde liegende Wortdatei
Abkuerzung.Word	Abkürzungen	PrepareData/PrepareAbkuerzung.xml	WordData/Abkuerzungen.ssd
Abkuerzung.Parameter	Ssd-Zeichenkette für entsprechende Abkürzungen		
Adjective.Positiv	Adjektive im Positiv	PrepareData/PrepareAdjektiv.xml	Nebsy/Adj.DAT
Adjective.Komparativ	Adjektive im Komparativ		
Adjective.Superlativ	Adjektive im Superlativ		
HilfsVerb.Word	Hilfsverben	PrepareData/PrepareHilfsVerb.xml	Nebsy/HVerb.DAT
HilfsVerb.Parameter	Grundformen der Hilfverben		
Konjunktion.Word	Konjunktionen	PrepareData/PrepareKonjunktion.xml	WordData/Konjunktion.ssd
Konjunktion.Parameter	Modus der Konjunktionen		
Numeral.Word	Numerale/Zahlenwörter	PrepareData/PrepareNumeral.xml	WordData/Numerale.ssd
Numeral.Parameter	ssd-Zeichenkette für entsprechende Numerale		
Praeposition.Word	Präpositionen	PrepareData/PreparePraeposition.xml	Nebsy/Praep.DAT
Praeposition.Parameter	ssd-Zeichenkette für entsprechende Präpositionen		
Pronom.Word	Pronomen	PrepareData/PreparePronom.xml	WordData/Pronomen.ssd
Pronom.Parameter	Modus der Pronomen		
Satzzeichen.Word	Satzzeichen	PrepareData/PrepareSatzzeichen.xml	WordData/Satzzeichen.ssd
Satzzeichen.Parameter	ssd- Zeichenkette für entsprechende Satzzeichen		
Adverb.Word	Adverbien	PrepareData/PrepareAdverb.xml	Nebsy/Adv.DAT
Adverb.	Modus der Adverbien		

Parameter			
Verben. Praefix	Präfixe von Verben	PrepareData/ Prepare Praefix.xml	Nebsy/ Praefix.DAT
Verben. Stamm	Stamm der Grundform von Verben	PrepareData/ Prepare Verb.xml	Nebsy/ Verb.DAT
Verben. Infinitiv	Grundformen von Verben		
Verben. Praesens	Präsens-Formen der Verben		
Verben. Praeteritum	Präteritum-Formen der Verben		
Verben. Perfekt	Perfekt-Formen der Verben		

Tabelle 6: Verarbeitung von Wort-Dateien

Die Wörter sind durch das Sonderzeichen „|“ getrennt in den Threadlist-Variablen mit ihren Variablennamen `Word` abgelegt.

```
Wort1|Wort2|Wort3
```

Bestimmte Wortarten - wie Konjunktionen, Pronomen, Adverben und Hilfsverben - haben ein Attribut, das in der Threadlist-Variablen mit den Variablennamen `.Funktion` der Reihe nach angelegt ist:

```
Attribute1|Attribute2|Attribute3
```

Die Wortarten, die die ssd-Zeichenkette für die entsprechende Wortart enthalten, sind in der Threadlist-Variablen mit dem Variablennamen `.Parameter` mit seinen Attributen als Zeichenkette in der ssd-Notation abgelegt:

Beispiel:

```
Personalpronomen: {Wort: "ich", Kasus: "Nom.", Person: "1. Pers."
Numerus: "Sing.", Genus : "" },
```

Bei Eingabe eines Satzes wird IPEE mit der Threadlist `SentenceTossd/WordAnalyse/WordAnalyse.xml` aufgerufen.

Dies geschieht mit der Methode `run()`, der von SentenceView angelegten Instanz `wordAnaylseThreadList` von der Klasse `ThreadList`, die eine Wort-Analyse

ausführt.

Die Wort-Analyse ist funktionell sehr ähnlich aufgebaut, wie die innerhalb der Arbeiten von Dischert und Kiefel[70] und Johann[71], so dass sich eine genauere Erklärung des Ablaufs hier erübrigt.
Grundsätzlich wird mit Hilfe der Wort-Analyse der Satz in einzelne Wörter zerlegt, und jedes einzelne Wort durchläuft den gesamten Analysezyklus. Dabei wird für jedes Wort die Groß- und Kleinschreibung berücksichtigt.
Wenn es klein geschrieben ist, wird keine Erkennung für das Substantiv durchgeführt; wird es groß geschrieben, folgt diese.
Beim ersten Wort eines Satzes (Satzanfang) sieht das Programm vor, die Analyse sowohl für Substantive als auch für klein geschriebene, einfache Wörter und klein geschriebene, flektierte Wörter durchzuführen.

10.2 Erkennung einfacher Wörter

Um das Verständnis auch für ein folgendes komplexeres Analyseverfahren des Verbs zu verdeutlichen, soll zunächst kurz auf die Analyse eines einfachen Wortes eingegangen werden:

Grundsätzlich wird mit Hilfe der Wort-Analyse der Satz in einzelne Wörter zerlegt und jedes einzelne Wort analysiert.

Die Analyse einfacher Wörter bzw. Analysen, denen Wortdateien zugrunde liegen, werden durch die Threadlist-XML-Datei `SentenceTossd/WordAnalyse/WordAnalyseData.xml` durchgeführt.

Grundsätzlich wird mit Hilfe des WordTypePattern (`SentenceTossd/WordAnalysePatterns/WordTypePattern.xml`) erkannt, ob eine bestimmte, einfache Wortart vorliegt.

Innerhalb der `WordAnalyseData.xml` Datei wird das WordTypePattern in Datei folgendermaßen aufgerufen:

```
<wordextractor patternlist="WordTypePattern" in=".Word"
invar=".Praeposition" out=".Result">
```

Mit dem Tupel `in=".Word"` wird das Wort als zu analysierender Fließtext übergeben und mit `invar=".Praeposition"` wird die Threadlist-Variable `.Praeposition`

70 Dischert/Kiefel, S. 42-61
71 Johann, S. 24-42

übergeben. In dieser sind in dem Variablennamen `Praeposition.Word` alle Modi der Präpositionen enthalten. (Der Inhalt dieser basiert auf den im Datensatz `Nebsy/Praep.DAT` enthaltenen Präpositionen.)

Das `WordTypePattern` sieht folgendermaßen aus:

```
...
<patternlist>
  <set>
    <item name="Parameter" transform="string" />
  </set>
  <pattern>
    <contents chars=" "/>
     <indexlist matchvalues="Word" returnvalues="Parameter"
           item="Parameter"/>
     <attribute>
       <synonym chars=" " separation="no"/>
     </attribute>
  </pattern>
</patternlist>
...
```

Innerhalb des Datensatzes des `SetElementes` befindet sich das `itemElement` mit dem Namen `Parameter`.

Das Pattern selber wird durch das Synonym mit dem `attribute chars=" "` ausgelöst, das dem `AttributeElement` untergeordnet ist.

Das Attribut `separation="no"` ist aus kundenspezifischen Gründen entstanden. Es bedeutet keine weitere Analyse der rechts oder links liegenden Zeichen innerhalb der Bearbeitung des Synonyms.

Führt das Pattern zu einem erfolgreichen Ergebnis, werden die weiteren untergeordneten Elemente des Patterns aufgerufen; in diesem Fall liegen auf der linken Seite (innerhalb der XML-Patternlist-Datei sind es die über dem Attribut Element liegenden) des Attribute-Elementes nicht weitere Elemente, die aufgerufen und zu einem wahren Ergebnis führen müssen.

Das folgende `IndexlistElement` vergleicht alle Zeichenketten, die in der Threadlist-Variablen `.Praeposition` mit dem Variablennamen `.Word` zu finden sind. (Die Threadlist-Variable `.Praeposition` wird hier nicht explizit angegeben, diese wurde beim Aufruf des Patterns mit dem Attribute `.invar` übergeben.)

```
<indexlist matchvalues="Word" returnvalues="Parameter"
item="Parameter"/>
```

Sollte eine Zeichenkette mit dem übergebenen Fließtext, der das zu analysierende Wort enthält, das links neben dem angefügten Leerzeichen steht, mit dem Wort übereinstimmen, so wird das Element aus der Threadlist-Variablen `.Praeposition` mit dem Namen `Parameter` übergeben, das an der gleichen Position steht. In diesem Fall sind das die Modi der aufgefundenen Präposition.

Das folgende Element führt zu einem wahren Ergebnis, wenn innerhalb des Fließtexts links neben den bis jetzt erkannten Zeichen nun ein Leerzeichen steht.

```
<contents chars=" "/>
```

Wenn der Pattern-Matching-Vorgang erfolgreich ist, wird in die Threadlist-Variable `.Result` das Attribut des erkannten Wortes geschrieben. In diesem Fall ist das der Modus der Präposition. Bei nicht erfolgreichem Matchen steht in dieser Threadlist-Variablen die Zeichenkette „`not matched`„.

Der Inhalt des folgenden `IfElementes` wird ausgeführt, wenn das Ergebnis nicht zu einem „`not matched`“ geführt hat, also wenn das Wort erkannt wurde:

```
   <if operation="notequal" operant1="Result.Parameter"
operant2="not matched">
```

Dieser sorgt dafür, dass das Ergebnis des Matching-Vorgangs entsprechend in das ssd-Format konvertiert und in die Variable `.ssdWords` hineingeschrieben wird.

Diese Verfahren sind für die jeweiligen Wortarten unterschiedlich, basieren aber grundsätzlich auf einfache Zusammensetzungen von Strings und sollen deswegen nicht weiter erläutert werden.

10.3 Erkennung eines unregelmäßigen Verbs

Innerhalb der Threadlist-XML-Datei `SentenceTossd/WordAnalyseWissen,` die zur Erkennung von flektierten, klein geschriebenen Wörter dient, wird der `WordExtraktor` folgendermaßen aufgerufen:

```
<wordextractor patternlist="VerbPattern" in=".Word" invar=".Verben"
out=".Result"/>
```

Es wird die Patternlist `VerbPattern` aufgerufen, die vorher innerhalb der Ausführung der

Threadlist `SentenceTossd/PrepareData/PrepareWordAnalyse.xml` angelegt wurde. Die Patternlist-XML-Datei hat den Verzeichnispfad `SentenceTossd/WordAnalysePatterns/VerbPattern.xml`

Mit dem Attribut `in` wird das zu analysierende Wort übergeben. Dieses wurde zuvor mit einem Leerzeichen vor und hinter der Zeichenkette des Wortes versehen.
Die Threadlist-Variable „`.invar`" enthält die Variablennamen, die die Zeitformen der unregelmäßigen Verben enthalten.

```
Stamm
Infinitiv
Praesens
Praeteritum
Perfekt
```

In der Threadlist-Variablen `.Result` sind die Variablennamen des ausgehenden Resultates enthalten.
Diese sind innerhalb des `SetElements` der Patternlist angegeben:

```
...
<set>
  <item name="Praefix" transform="string"/>
  <item name="Infinitiv" transform="string"/>
  <item name="Person" transform="string"/>
  <item name="Zeit" transform="string"/>
</set>
...
```

Für die Zeiten Präsens, Perfekt und Präteritum existieren in dieser Patternlist-XML-Datei drei weitere Patterns, die konzeptionell ähnlich aufgebaut sind und deswegen nicht weiter erläutert werden. Beim Aufruf des `Wordextractors` werden gleichzeitig alle Patterns aufgerufen, deren Synonyme eine Übereinstimmung mit der Endung des übergebenen Wortes haben.

Aufgerufen wird das Pattern durch den `Wordextractor` an den Synonym-Elementen der übergeordneten Attribute-Elemente:

```
...
<pattern>
  <contents chars=" "/>
  <pattern insert="PraefixPattern"/>
  <setvar value="Praesens" item="Zeit"/>
  <xor>
   <indexlist matchvalues="Stamm" returnvalues="Infinitiv"
         item="Infinitiv"/>
   <indexlist matchvalues="Praesens" returnvalues="Infinitiv"
    item="Infinitiv"/>
```

```
   </xor>
   <xor item="Person">
    <attribute value="Imperativ Sing." var="string"
onlymatched="no">
    <synonym chars=" " separation="no"/>
    </attribute>
    <attribute value="2. Pers. Sing." var="string" onlymatched="no">
    <synonym chars="st " separation="no"/>
    </attribute>
    <attribute value="3. Pers. Sing./2. Pers. Plu./Imperativ Plu."
     var="string" onlymatched="no">
    <synonym chars="t " separation="no"/>
    </attribute>
   </xor>
 </pattern>
 ...
```

Aufgerufen wird das Pattern durch den `Wordextractor` anhand der `SynonymElemente`, die den `AttributeElementen` untergeordnet sind:

```
...
<attribute value="Imperativ Sing." var="string" onlymatched="no">
     <synonym chars=" " separation="no"/>
</attribute>

<attribute value="3. Pers. Sing./2. Pers. Plu./Imperativ
     <synonym chars="t " separation="no"/>
</attribute>

<attribute value="2. Pers. Sing." var="string" onlymatched="no">
     <synonym chars="st " separation="no"/>
</attribute>
...
```

Diese `SynonymElemente` haben die Endungen „ „, „t „ und „st „, die durch die Attribute `chars` beschrieben werden. Dass dies nur Endungen sein können, wird durch das Leerzeichen am Ende ausgedrückt. Das zu analysierende Wort enthält ebenfalls ein Leerzeichen am Ende.

Das Attribute-Element ist durch ein `XorElement` umhüllt, das das `item="Person"` enthält. Das bedeutet, dass nur eins von den drei Attribute-Elementen erkannt werden muss, damit das `XorElement` erfüllt ist.

Bei erfolgreicher Erkennung des gesamten Patterns werden die `AttributeElemente`, die durch das Attribut „`value`“ angegeben sind, in das Item `Person` des ausgehenden Datensatzes geschrieben.

Das Pattern ruft nun sein nächstes linkes Element (innerhalb der Ansicht des XML-Dokuments: das nächst oberste Element). Dies ist ein `XorElements`, das zwei

IndexlistElemente enthält.

```
...
<xor>
  <indexlist matchvalues="Stamm" returnvalues="Infinitiv"
     item="Infinitiv"/>
  <indexlist matchvalues="Praesens" returnvalues="Infinitiv"
     item="Infinitiv"/>
</xor>
...
```

Das erste IndexlistElemente vergleicht nun den Stamm des Wortes mit allen Einträgen in den Variablen, die mit dem Variablennamen „Stamm“, die innerhalb der Threadlist mit dem Attribut invar übergeben wurde, angelegt sind.
Sollte eine Übereinstimmung erkannt werden, so wird durch das Attribut returnvalues="Grundform" das Element aus Grundform zurückgegeben, das die gleiche Position in der Liste enthält, wie die gefundene Zeichenkette in „Stamm“. Diese wird dem „item“ Grundform übergeben.
Sollte keine erfolgreiche Erkennung stattfinden, so ruft das XorElement auch das IndexListElement auf mit dem matchvalues="Praesens", da die erkannten grammatischen Personen und Zeiten sowohl aus dem Stamm der Grundform als auch aus dem Stamm der ersten Person Präsens gebildet werden können (siehe Kapitel 3.2.3 „Verben“).

Das nächste Element SetvarElement schreibt den Wert „Praesens“ in das Item „Zeit“:

```
<setvar value="Praesens" item="Zeit"/>
```

Das folgende Pattern-Element ruft ein untergeordnetes Pattern auf mit dem Namen "PraefixPattern", das am Ende der Patternlist-Datei zu finden ist:

Aufrufendes PatternElement:

```
<pattern insert="PraefixPattern"/>
```

Das PraefixPattern:

```
<pattern name="PraefixPattern">
  <xor>
    <indexlist matchvalues="Praefix" returnvalues="Praefix"
           item="Praefix"/>
    <setvar value="no" item="Praefix"/>
  </xor>
```

```
</pattern>
```

Diese Patternlist prüft, ob am Anfang des Verbs noch ein Präfix zu finden ist (z. B. das Verb *gehen* mit dem Präfix *fort=fortgehen); a*nsonsten wird in Präfix die Zeichenfolge „no“ geschrieben.

Wenn alle Elemente des Patterns ausgeführt werden können, gilt das Pattern als erfolgreich gematcht. Nur in diesem Fall werden auch alle erfassten Variablen in die entsprechenden Item-Elemente geschrieben.

Nachdem das Wort-Extraktions-Verfahren durchgeführt wurde, wird die Threadlist weiter bearbeitet. Darauf folgt das Element:

```
<include
file="SentenceTossd/WordAnalyse/Concat/VerbinResult.xml"/>
```

Wenn der Matchingvorgang erfolgreich war, konvertiert die Threadlist-XML-Datei `SentenceTossd/WordAnalyse/Concat/VerbinResult.xml` den ausgehenden Datensatz der durchgeführten Extraktion in das ssd-Format und schreibt das Ergebnis in die Threadlist-Variable „`WordAnalyse`“.

10.4 Erkennung regelmäßiger Verben

Innerhalb der unregelmäßigen Verben werden die Stämme anhand von übergebenen Listen verglichen. Anders ist es jedoch bei den regelmäßigen Verben. Diese erkennen zwar ebenfalls die Endungen mit Hilfe von `Synonym`-Elementen, jedoch wird kein Vergleich der Stämme durch `IndexlistElement` durchgeführt, sondern der mögliche Stamm wird von der Endung abgetrennt und in das Item `Grundform` des ausgegebenen Datensatzes geschrieben. Dies geschieht mit folgendem Element:

```
<contents regex="[\w|ö|ä|ü|Ö|Ä|Ü|ß]+" var="string"
item="Grundform"/>
```

Das Attribut „regex“ gibt einen regulären Ausdruck wieder[72], der alle Zeichen von a-z und A-Z erkennt, ebenso alle Sonderzeichen wie ö ä ü Ö Ä Ü ß und diese in das Item der Grundform schreibt.
Wird eine Endung erkannt, wird der Stamm mit Hilfe einer Internet-Recherche auf das Lexikon wissen.de überprüft.

72 Sun Microsystems: “java.util.regex Class Pattern“ http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html (2.2007)

10.5 Erkennung von Nominalphrasen

Für die Phrasen-Analyse wird IPEE mit der Threadlist-XML `SentenceTossd/PhraseAnalyse/PhraseAnalyse.xml` aufgerufen.

Dies geschieht mit der Methode `run()`, der von SentenceView angelegten Instanz `phraseAnaylseThreadList` von der Klasse `ThreadList`.

Innerhalb dieser Threadlist-XML wird eine Nominalphrase und darauf eine Präpositionalphrasen-Erkennung durchgeführt. Es soll hier nur auf das Pattern für die Nominalphrasen-Erkennung eingegangen werden.

Im Gegensatz zu dem von Dischert und Kiefel vorgestellten Verfahren, das mehrere Muster für die Nominalphrasen-Erkennung anwendet[73], sind hier die Muster in einem Pattern integriert (im folgenden „Nominalphrasen-Pattern“ genannt), das weitere Unterpatterns hat, die ausschließlich für die Erkennung der einzelnen Wortarten verantwortlich sind. Diese Nominalphrasen-Pattern und die notwendigen Unterpatterns sind in der Threadlist-XML-Datei `SentenceTossd/PhraseAnalyse/NominalphrasePattern.xml` wiederzufinden.
Das Nominalphrasen-Pattern hat auf der ersten Ebene sechs untergeordnete Elemente, die sich kommentiert folgendermaßen beschreiben lassen (die Reihenfolge ist mit der Anordnung innerhalb des Patterns identisch):

- Erkennung des Satzteiles vor der Nominalphrase
- Erkennung einer Nominalphrase, die mit einer Kardinalzahl oder Zahl beginnt
- Erkennung einer Nominalphrase mit Artikel, Possessivpronomen oder Demonstrativpronomen und möglichen, folgenden Numeralen, Zahlenwörtern oder Zahlen
- Erkennung der Adjektive und Partizipien (kann auch als Erkennung der Adjektivphrase betrachtete werden)
- Erkennung des Substantivs
- Erkennung des Satzteils nach der Nominalphrase

Bis auf die Erkennung des Substantivs sind alle diese beschriebenen, untergeordneten Patterns mit einem `MayElement` eingekleidet; d. h. diese können bei der Findung einer Nominalphrase zu einem erfolgreichen Matching-Vorgang führen. Dies ist jedoch kein Muss-Kriterium. Eine Nominalphrase kann auch mit nur einem Substantiv gebildet werden.

Für die Erkennung des Substantivs steht in dem untergeordneten Pattern das Element

```
<pattern insert="Substantiv"/>
```

Das Unterpattern mit dem Namen `"Substantiv"` ist folgendermaßen definiert:

73 Dischert/Kiefel, S. 62 - 64

```
...
<pattern name="Substantiv">
   <attribute var="string" value="Substantiv:"
item="SubstantivWordType"  onlymatched="no">
     <synonym chars="Substantiv:" separation="no"/>
   </attribute>

   <pattern insert="subContent" item="SubstantivAttribute"/>

</pattern>
...
```

Mit dem Element `<pattern insert="subContent" item="SubstantivAttribute"/>` werden mittels eines weiteren Unterpatterns mit dem Namen `subContent` die Attribut-Werte-Paare des Substantivs erkannt und in das Item `SubstantivAttribute` geschrieben.
Für alle weiteren Wortarten (wie Artikel, Adjektive, Partizipien, Kardinalzahlen, Ordnungszahlen, Zahlwörter, Zahlen, Satzzeichen, Possessivpronomen, Demonstrativpronomen und Konjunktion), die innerhalb der Nominalphrase vorkommen, existiert jeweils ein Unterpattern, das dem Aufbau des Patterns für die Substantiv-Erkennung konzeptionell ähnelt.
Diese enthalten jedoch keine Attribute-Elemente (wie das Unterpattern) zur Erkennung des Substantivs und können somit nicht ausgelöst werden. Stattdessen enthalten sie ein einfaches `ContentsElement,` das zur Erkennung einer Zeichenkette (Attribute `chars`) dient und in dem das entsprechende Wort eingetragen ist.

```
...
<pattern name="Artikel">
   <contents chars="Artikel:" var="string"
     item="DiscribingWordTypes"/>
   <pattern insert="subContent" item="DiscribingAttributes"/>
</pattern>
...
```

Das Nominalphrasen-Pattern wird nun erst einmal ab dem erkennenden Element für das Substantiv von links nach rechts fortgesetzt. Das rechts liegende, untergeordnete Element erkennt die Satzelemente nach dem Pattern.
Darauf wird der linke Teil vor dem Element des Nominalphrasenpatterns, das das Substantiv erkennt, von rechts nach links bearbeitet.

Das folgende Element erkennt eine Anreihung von Adjektiven oder Partizipien bzw. die untergeordnete Adjektivphrase der Nominalphrase. Die Adjektive oder Partizipien können mit einer bestimmten Regel durch eine Konjunktion, einem Satzzeichen oder auch ohne ein weiteres Element erkannt werden, z. B. „fahrende, schnelle“, „gute, schnelle“ „fahrende, gute und schnelle“, „gute und schnelle“.
Das Pattern wird ebenfalls von rechts nach links abgearbeitet. Wenn zwei Adjektive oder

Partizipien durch eine Konjunktion getrennt sein sollten, sind diese grundsätzlich am Schluss (rechts) der Adjektivphrase aufzufinden: folgende Unterelemente erkennen entweder ein Adjektiv oder ein Partizip, wenn vorher eine Konjunktion steht:

```
...
<pattern insert="Partizip"/>
  <may>
    <and>
      <pattern insert="Konjunktion"/>
      <xor>
        <pattern insert="Adjektiv"/>
        <pattern insert="Partizip"/>
      </xor>
    </and>
 </may>
...
```

Liegt keine Konjunktion vor, so wird dieses durch das `MayElement` eingekleidete Unterelement übersprungen.
Die links neben der Konjunktion (wenn vorhanden) liegenden Adjektive oder Partizipien sind möglicherweise durch ein Satzzeichen „,“ getrennt und werden durch folgende Unter-Elemente erkannt:

```
...
<may>
  <or>
    <and>
      <may>
        <pattern insert="Satzzeichen"/>
      </may>
      <xor>
        <pattern insert="Adjektiv"/>
        <pattern insert="Partizip"/>
      </xor>
    </and>
  </or>
</may>
...
```

Das `OrElement` unter dem `MayElement` auf der obersten Ebene sorgt dafür, dass die Erkennung eines Adjektivs oder Partizips iterierend erfolgt.

Dazwischen wird entweder ein Adjektiv oder eine Partizip erkannt (`XorElement`), das evtl. durch ein Satzzeichen getrennt ist (`MayElement` oberhalb des `<pattern insert="Satzzeichen"/>` Elementes).

Eine Nominalphrase kann vor einer möglichen Adjektivphrase durch einen Artikel, ein

Possessiv- oder Demonstrativpronomen eingeleitet sein.
Das nächste dem gesamten Pattern untergeordnete Element ist wiederum durch ein `MayElement` und ein `AndElement` (das die Reihenfolge wiedergibt) eingekleidet:

```
...
<xor>
  <pattern insert="Artikel"/>
  <pattern insert="Possessivpronomen"/>
  <pattern insert="Demonstrativpronomen"/>
</xor>
...
```

Wenn Zahlwörter folgen, wie eine Kardinalzahl, eine Ordnungszahl, eine Zahl oder ein Zahlwort, setzt sich das Pattern folgendermaßen fort:

```
...
<may>
  <xor>
    <pattern insert="Kardinalzahl"/>
    <pattern insert="Ordnungszahl"/>
    <pattern insert="Zahl"/>
    <pattern insert="Zahlenwort"/>
  </xor>
</may>
...
```

Das folgende erste der Nominalphrasen-Pattern untergeordnete Element dient zur Erkennung von Zahlwörtern von Nominalphrasen, die entsprechend eingeleitet sind: Z.B. „drei Tassen“, „ 40 Tassen“, „alle drei Tassen“.

```
...
<may>
  <or>
    <pattern insert="Kardinalzahl"/>
    <pattern insert="Zahl"/>
    <and>
      <pattern insert="Zahlenwort"/>
      <may>
        <or>
          <pattern insert="Kardinalzahl"/>
          <pattern insert="Zahl"/>
        </or>
      </may>
    </and>
  </or>
</may>
...
```

Schlussendlich werden mit dem ersten untergeordneten Element des Nominalphrasen-Patterns die Wörter vor der Nominalphrase erfasst.

11 Semantisch orientierte Verarbeitung

Nachdem vorangehend die semistrukturierte Konzeption und der Einsatz innerhalb der Wort- und Phrasen-Analyse mit Hilfe der Softwarelösung IPEE beschrieben wurde, soll in diesem Kapitel auf das wesentliche Ziel der semantisch orientierten Verarbeitung eingegangen werden.

Das Fazit des ersten Kapitels ist, dass die Semantik einer Information dadurch entsteht, dass eine Information mit schon vorhandenem Wissen des verarbeitenden Systems in Beziehung gesetzt werden kann. Dieses In-Beziehung-Setzen ermöglicht Vorgänge wie Vergleichen, Verbinden, Trennen und Abstrahieren.

Ein semantisch orientiertes Verarbeitungs-System, das versucht einige Fähigkeiten unseres menschlichen Geistes nachzubilden, ist komplex. Deswegen wurde in dieser Masterarbeit beschlossen, sich auf die Verarbeitung von Verben zu beschränken.

Das Konzept zur semantisch orientierten Verarbeitung, das hier vorgestellt wird, basiert darauf, dass ein eingegebener Satz abstrahiert wird, - in dieser Masterarbeit lediglich das jeweilige Verb. Die Verben und die Abstraktion werden miteinander verglichen. Stimmt ein Verb oder eine Abstraktion überein, so werden diese verbunden; ansonsten bleiben sie getrennt.

Es stellt sich nun die Frage, wie innerhalb dieser Verfahren der Begriff Abstraktion verstanden werden soll. Dazu soll der Begriff Abstraktion aus der Philosophie folgendermaßen definiert werden: das Absehen von bestimmten Aspekten der konkreten, miteinander zusammenhängenden Dinge, Personen oder Vorgänge in Natur und Gesellschaft.

Somit soll der Begriff abstrahieren/Abstraktion als Vorgang betrachtet werden, der von einem konkreten Element zugehörige Aspekte trennt und mit einem weiteren Element gleichsetzt.

Für die Abstraktion des Verbs sollen im folgenden zwei Verfahren vorgestellt werden:

Im ersten Verfahren werden für das Verb des eingegebenen Satzes Bedeutungserklärungen recherchiert. Das Verb innerhalb der Bedeutungserklärung stellt einen Oberbegriff (Hyperonym) [74] des Verbs des eingegebenen Satzes dar.

Im zweiten Verfahren soll das Verb einem Verbfeld oder Makrofeld zugeordnet werden, die innerhalb des Valenzwörterbuchs „Verben in Feldern“ definiert sind und jeweils eine bestimmte Menge von Verben enthalten.

Das semantisch In-Beziehung-Setzen entsteht in diesen beiden Verfahren dadurch, dass

74 Oberbegriff: Begriff, der eine Anzahl anderer Begriffe in ihrer Bedeutung repräsentiert.

zwei Elemente mit einem übergeordneten Element in Verbindung gesetzt werden.
Das übergeordnete Element ist im ersten Verfahren ein dem Verb zugeordnetes Hyperonym und im zweiten Verfahren die Zuordnung eines Verbfeldes bzw. Makrofeldes.

11.1 Zuordnung eines Hyperonyms zum Verb durch Recherche aus dem Wörterbuch (wissen.de)

Das erkannte Verb des Satzes wird in einem Wörterbuch recherchiert, und die Bedeutungen des Verbs werden mit Hilfe des vorgestellten Wort- und Phrasen-Analysesystems analysiert.
In dem realisierten Verfahren wird bei der Recherche eines Verbs auf das Internet-Lexikon wissen.de zugegriffen, das Einträge aus dem BERTELSMANN Wörterbuch der Deutschen Sprache enthält.

Innerhalb dieses Wörterbuchs sind für ein Verb Lesarten eingetragen, die in diesem Dokument mit folgender Notation beschrieben sind:
Notation von Lesarten: { Grammatische Angaben; Bedeutungsnummer; Bedeutungserklärung; weitere Angaben zur Bedeutungserklärung (Beispiele) }
Für folgende Betrachtungen nicht relevante Elemente innerhalb der Lesarten werden mit „...“ dargestellt.

Beispiel-Satz: *Ich fahre mit dem Auto in die Stadt.*

Für das Verb *fahren* sind mehrere Lesarten im Wörterbuch zu finden:

{ ...; I.1; *sich auf Rädern oder durch Triebkraft fortbewegen*; ... }
{ ...; I.2; *sich auf, in einem Fahrzeug fortbewegen*; ... }
{ ...; I.3; *ein Fahrzeug lenken*; ... }
{ ...; I.4; *sich gleitend, sanft oder schnell bewegen*; ... }
etc..[75]

Das Verb *fahren* hat die Hyperonyme *(fort)bewegen* und l*enken*. Das bedeutet: Wörter wie *(fort)bewegen* und *lenken* haben in Bezug zum Verb *fahren* eine semantisch ähnliche Aussage.

Ähnlich verhält es sich mit dem Verb *fliegen*:

{ ...; I.1; *sich aus eigener Kraft mit Flügeln durch die Luft bewegen*; ... }
{ ...; I.2; *sich gleitend, sanft oder schnell bewegen*; ... }
{ ...; I.3; *sich durch mechanische Auftriebskraft durch die Luft bewegen*; ... }
{ ...; I.4; *sich mit einem Luftfahrzeug fortbewegen, mit dem Flugzeug reisen*; ... }
{ ...; II.1; *in der Luft führen, lenken*; ... }
etc.....[76]

75 wissen.de: Schlagwort: fahren, http://www.wissen.de/wde/generator/wissen/services/suche/wbger/index.html?gerqry=fahren (2.2007)
76 wissen.de: Schlagwort: fliegen,

Das Verb *fliegen* hat die Hyperonyme *(fort)bewegen, reisen, führen, lenken.*
Somit haben die Verben *fahren* als auch *fliegen* die Hyperonyme *(fort)bewegen* und *lenken.*
d. h. die Verben *fahren* und *fliegen* haben eine semantische Übereinstimmung.

11.2 Zuordnung des Verbs eines Verbfeldes bzw. Makrofeldes mit Hilfe des Valenzwörterbuchs „Verben in Feldern“

Eine weitere Möglichkeit, dem Verb ein übergeordnetes Element zuzuordnen, um semantisch orientiert in Beziehung setzen zu können, bietet das Valenzwörterbuch „Verben in Feldern“, Hrsg. Helmut Schumacher.

„Dieses Wörterbuch ist ein Versuch, die Bedeutung von über 1000 Verben und verbalen Ausdrücken zu erklären und ihre valenzbestimmende Umgebung syntaktisch und semantisch zu charakterisieren.“[77]
„Die behandelten Verben sind durchgängig nach onomasiologischen Kriterien, d. h. aufgrund ihrer Bedeutungsverwandtschaft, hierarchisch gruppiert.“[78]

Das Valenzwörterbuch hält somit eine hierarchische Struktur bereit, die baumartig aufgebaut ist. Die Struktur kann maximal sechs Ebenen haben und zwar Makrofeld, Verbfeld, Subfeld, Verbgruppe, Subgruppe, Verb[79], wovon erst einmal die beiden obersten Ebenen und die unterste betrachtet werden sollen.
Das Wörterbuch umfasst sieben Makrofelder, die mit einstelligen Zahlen gekennzeichnet sind:

1. Verben der allgemeinen Existenz
2. Verben der speziellen Existenz
3. Verben der Differenz
4. Verben der Relation und des geistigen Handelns
5. Verben des Handlungsspielraums
6. Verben des sprachlichen Ausdrucks
7. Verben der vitalen Bedürfnisse[80]

In dem Makrofeld „Verben der allgemeinen Existenz“ sind folgende Verbfelder enthalten:

1.1.1 Zustandsverben der allgemeinen Existenz[81]
es gibt · existieren · bestehen
geschehen · sich ereignen · passieren · stattfinden

http://www.wissen.de/wde/generator/wissen/services/suche/wbger/index.html?gerqry=fliegen
77 Schumacher: „Verben in Feldern“, S. 1
78 Schumacher: „Verben in Feldern“, S. 9-10
79 Schumacher: „Verben in Feldern“, S. 10
80 Schumacher: „Verben in Feldern“, S. 12
81 Schumacher: „Verben in Feldern“, S. 71

1.2.1 Vorgangsverben der allgemeinen Existenz
es kommt zu · zustande kommen · sich entwickeln · entstehen · sich bilden · aufkommen · sich konstituieren · sich ausbilden · kommen zu · dazukommen · hinzukommen [82]

Die Abstraktion des Verbs erfolgt dadurch, dass das Verb einem Verbfeld zugeordnet wird, welches noch weiter zu einem Makrofeld abstrahiert werden kann.

11.3 Realisierung des semantisch In-Beziehung-Setzens von Verben

Praktisch realisiert wurde das Verfahren in Kapitel 11.1 „Zuordnung eines Hyperonyms zum Verb durch Recherche aus dem Wörterbuch (wissen.de)“ durch Recherche aus dem Wörterbuch (wissen.de). Mit Hilfe des im folgenden beschriebenen Verfahrens wird ein OEM-Graph (im folgenden Verbfeld-Graph genannt) generiert, der eine Wurzel enthält, der die einzelnen in das System eingegebenen Sätze untergeordnet werden. Den Verben der Sätze werden wiederum die Bedeutungen des Verbs untergeordnet.

Die semantische Übereinstimmung von zwei eingegebenen Sätzen wird folgendermaßen zum Ausdruck gebracht: Identische Verben werden miteinander durch eine Quer-Referenz (siehe Kapitel 2.4.1 „Die semistrukturierte Notation ssd“) verbunden. Z. B. bei Eingabe der beiden Sätze *Ich fahre mit dem Auto in die Stadt* und *Ich fliege mit dem Flugzeug über die Berge* und Recherche der Bedeutungen, werden die Verben *(fort)bewegen* und *lenken* verbunden.

Im folgenden soll das Verfahren zur Generierung des Verbfeld-Graphen beschrieben werden:
Nach Eingabe eines Satzes und Aktivierung der Verbfeld-Graph-Generierung
wird eine Wort- und Phrasen-Analyse durchgeführt und in das ssd-Format umgewandelt. Die Verben werden herausgenommen, und die Bedeutungen werden im Internet-Lexikon wissen.de recherchiert. Aus dem Webinhalt werden diese extrahiert und ebenfalls mit Hilfe der Wort- und Phrasen-Analyse in eine ssd-Notation überführt. Diese Bedeutungen werden mit einem Unterknoten mit der Referenz `Bedeutung` dem Verb angehängt.

Ein Beispiel ist im OEM-Graph in Abbildung 5 dargestellt, der eine Erweiterung zu Abbildung 6 in Kapitel 5 „Anwendung der Konzeption semistrukturierter Daten für Satzmuster“ bietet, und dem der Satz *„Ich fahre mit dem Auto“* zugrunde liegt.

Beim ersten Ausführen der Methode für die Verbfeld-Graph-Generierung entsteht zunächst ein Baum, der eine Wurzel (root im OEM-Graph: &r) hat, die mit einem weiteren Knoten mit dem Attribut „Satz“ referenziert .

(Innerhalb des OEM-Graphen in Abbildung 5 sind aus Übersichtlichkeitsgründen alle

82 Schumacher: „Verben in Feldern“, S. 80

Objekte, außer den atomaren Objekten, die keine Referenz mit dem Attribut „Wort“ haben, eingezeichnet.)

Die Elemente, die ein Wort repräsentieren, werden mit dem Attributnamen, die den Wortarten entsprechen, referenziert. Ebenso werden die Elemente, die eine Phrase (wie Nominal- oder Präpositionalphrase) wiedergeben, nach ihrer Bezeichnung referenziert.

Dem Verb *fahren* ist nun einem Element der eingehenden Referenz „Bedeutung“ untergeordnet.
Dieses Element ist wiederum den Elementen der wort- und phrasen-analysierten Bedeutungserklärung *sich auf Rädern oder durch Triebkraft fortbewegen* aus der Lesart mit Bedeutungsnummer I.1 des Verbs *fahren* aus wissen.de[83] untergeordnet.

83 wissen.de: Schlagwort: „fahren“ http://www.wissen.de/wde/generator/wissen/services/suche/wbger/index.html?gerqry=fahren (2.2007)

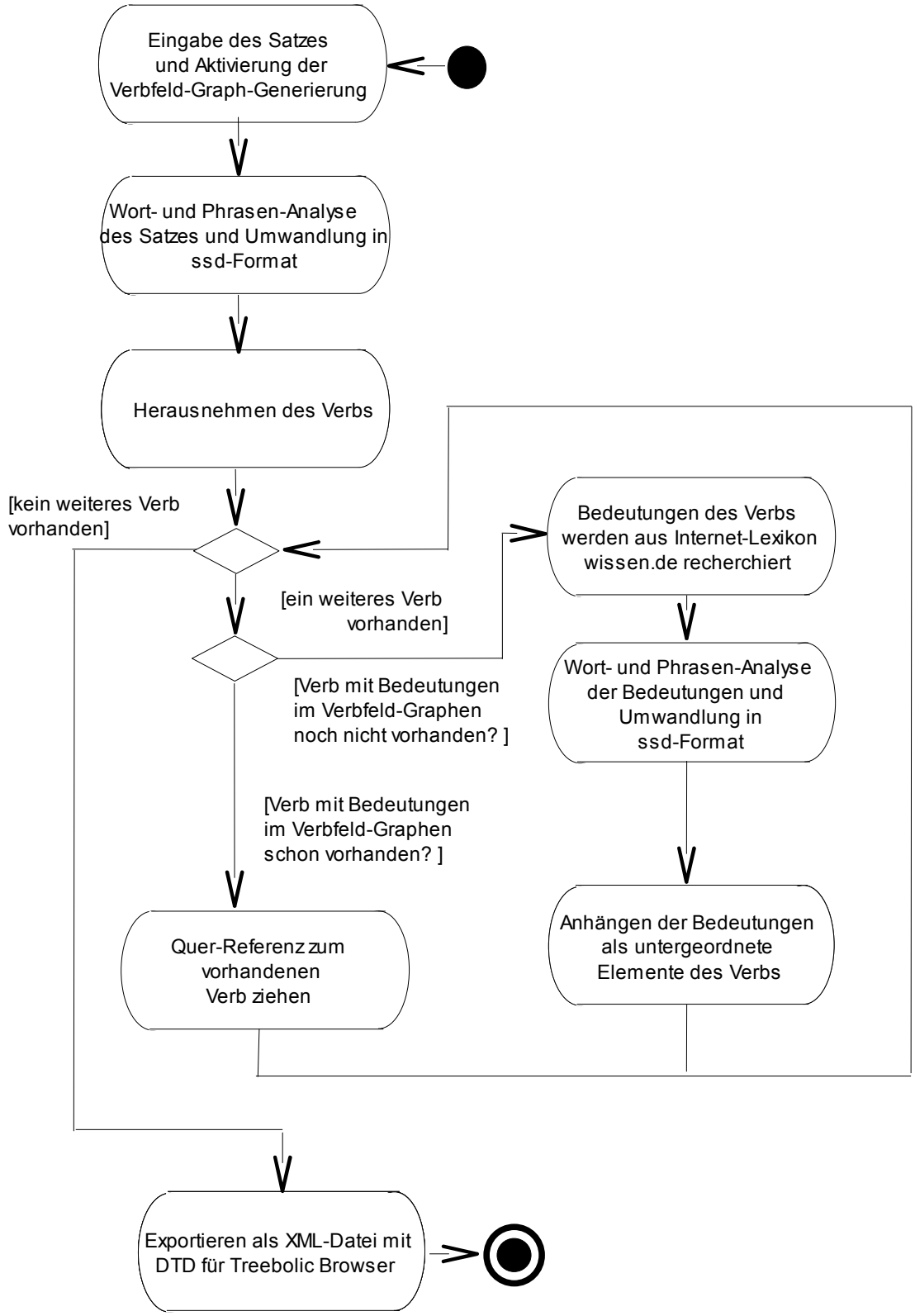


Abbildung 9: UML-Aktivitätsdiagramm zur Generierung eines Verbfeld-Graphen

Nach dem ersten Ablauf dieser Methode kann ein weiterer Satz eingegeben werden, der

wiederum mit dem gleichen Verfahren bearbeitet und ebenfalls der Wurzel des gesamten Baumes untergeordnet wird.

Wie schon beschrieben, stellen die Bedeutungen eine Abstraktion des Verbs dar.
Sollte bei Bearbeitung eines zweiten Satzes ein Verb mit angehängten Bedeutungen schon im Verbfeld-Graphen vorhanden sein, so wird dieses nicht nochmals mit Bedeutungen versehen. Es wird eine Quer-Referenz zwischen dem gerade bearbeiteten Verb und dem schon vorhandenen Verb gezogen, die die semantische Ähnlichkeit der Verben von beiden eingegebenen Sätzen darstellt. Durch diese entstandene Quer-Referenz kann nun nicht mehr von einem Baum gesprochen werden, sondern von einem gerichteten Graphen.

Dieser Graph wird als XML-Datei, der der DTD des Treebolic-Browsers unterworfen ist, exportiert und kann damit angesehen werden.

Das bis jetzt beschriebene Verfahren ist im UML-Aktivitätsdiagramm in Abbildung 14 skizziert, wobei die Aktivitäten auf funktionaler (nicht auf implementiertechnischer) Ebene dargestellt sind.

Nach diesem vorgestellten Verfahren enthält der Graph Elemente, die die eingegebenen Sätze beschreiben und die Bedeutungen des Verbs, die als Elemente dem Verb untergeordnet sind.

Das realisierte Verfahren sieht auch vor, mit rekursiven Ebenen zu arbeiten. Das heißt, dass die Bedeutungen wiederum analysiert werden und den Verben innerhalb der Bedeutungen wiederum Bedeutungen angehängt werden. Dazu wird innerhalb der Aktivität „Anhängen der Bedeutungen als untergeordnete Elemente des Verbs“ der Anwendungsfall „Herausnehmen des Verbs“ für die recherchierten Bedeutungserklärungen rekursiv aufgerufen.

Der entstehende Verbfeld-Graph wird als XML-Datei, die nach der DTD des Treebolic Browsers geformt ist, exportiert[84] und kann anschließend mit diesem Tool angesehen werden (siehe Kapitel 12 „Hyperbolische Bäume“).

Da ein Verb häufig mehrere (bis zu fünfzehn) Bedeutungserklärungen hat, entsteht dabei ein Graph mit einer unübersichtlichen Anzahl von Elementen. Für eine übersichtliche Visualisierung wird folgender Lösungsweg gegangen:

Es wird der Graph noch ein weiteres Mal exportiert mit einer reduzierten Menge an Knoten-Elementen.

Den Verben des reduzierten Graphen werden gegenüber dem ursprünglichen Graphen alle Elemente mit dem Referenznamen „Bedeutung“ abgeschnitten, deren untergeordnete Elemente keine Quer-Referenz aufweisen. Auf eine genauere Beschreibung des algorithmischen Verfahrens soll hier nicht eingegangen werden.

84 als Datei abspeichern

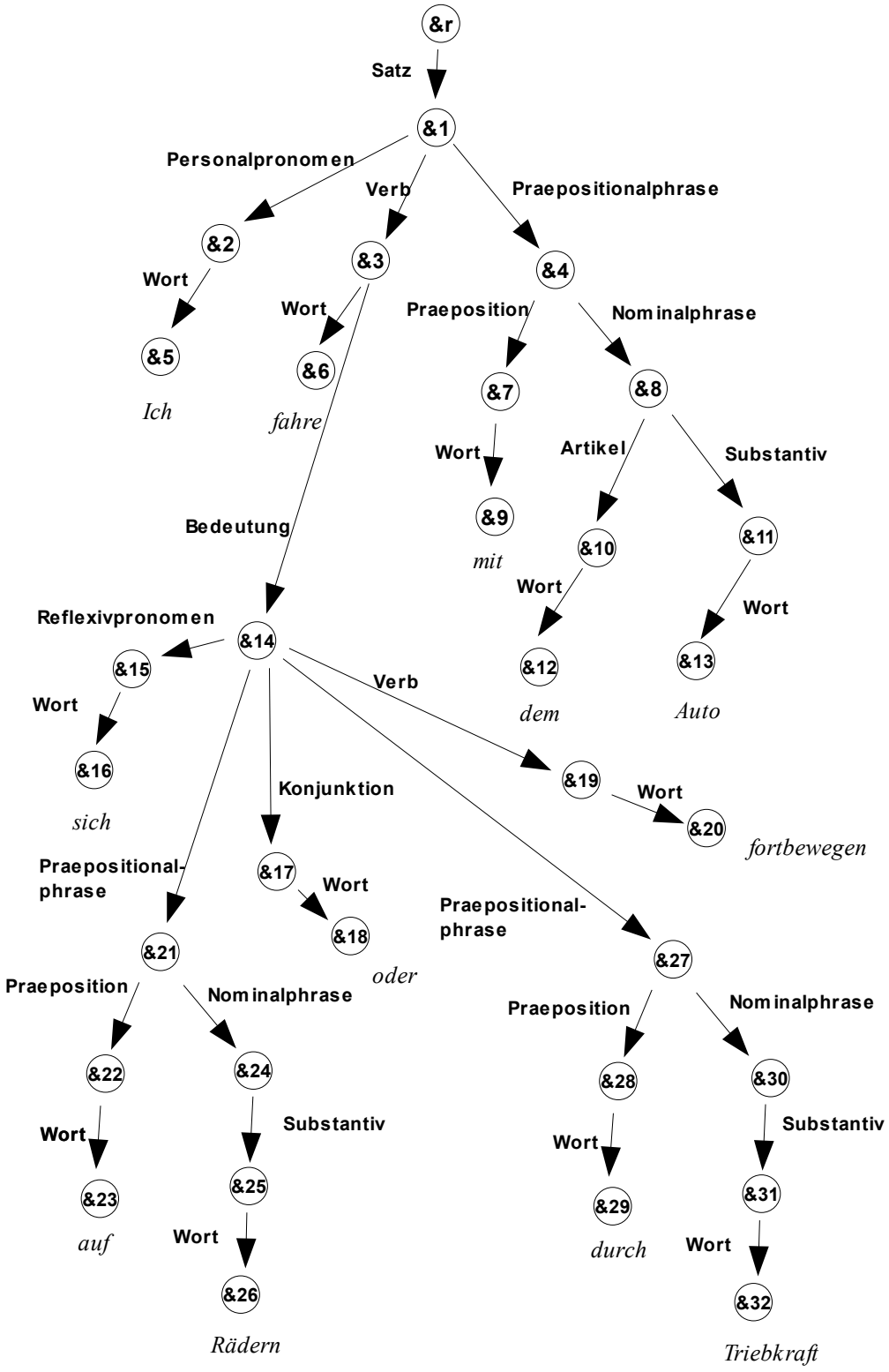


Abbildung 10: OEM-Graph von einem Beispielsatz mit Bedeutung

11.4 Experimentelle Auswertung der semantisch orientierten Bearbeitungsverfahren

Von Professor Dr. phil. Gregor Büchel wurde im Rahmen dieses Projektes eine eigene semantische Klassifikation von Verben (SKV)[85] erstellt, die ähnlich wie das Valenzwörterbuch „Verben in Feldern“ eine Anzahl von Verben einem Verbfeld oder Makrofeld (hier Klassifikation genannt) zuordnet.

Im folgenden soll nun untersucht werden, ob Verben, die der gleichen Klassifikation der SKV angehören, mit den beiden oben beschriebenen Verfahren in Beziehung setzbar sind.

Für das in Kapitel 11.1 gezeigte Verfahren „Zuordnung eines Hyperonyms zum Verb durch Recherche aus dem Wörterbuch (wissen.de)“ werden zwei oder mehrere Verben aus der gleichen Klassifikation in das System (siehe Kapitel 11.3 „Realisierung des semantisch In-Beziehung-Setzens von Verben“) eingegeben und entsprechend ein Verbfeld-Graph generiert.
Anhand des daraus entstehenden Verbfeld-Graphen kann nun erkannt werden (z. B. durch Betrachten des Graphen mit dem Treebolic Browser - siehe Kapitel 12 „Hyperbolische Bäume“), ob eine semantische Übereinstimmung durch Abstraktion der Verben stattgefunden hat.

Wenn dies der Fall ist, wird die gefundene semantische Beziehung in der Tabelle 7 „Semantisches In-Beziehung-Setzen von Verben„ (Spalte 3) folgendermaßen dargestellt:

(eingegebenes Verb1, eingegebenes Verb2) → abstrahiertes Verb

Für die Zuordnung eines Verbfeldes bzw. Makrofeldes eines Verbs anhand des Verfahrens „Zuordnung des Verbs eines Verbfeldes bzw. Makrofeldes mit Hilfe des Valenzwörterbuchs 'Verben in Feldern'“ (siehe Kapitel 11.2) sind in Spalte 2 der folgenden Tabelle das Makrofeld und das Verbfeld angegeben, wenn es im Valenzwörterbuch vorkommt.

85 Büchel: „Semantische Klassifikation von Verben (SKV)“, siehe Anhang B

semantische Klassifikation **Verben der/des ...**	**Verben aus der SKV** Die Klammerangaben entsprechen den Verbfeld-Zuordnungen aus dem Valenzwörterbuch „Verben in Feldern „ Notation (Makrofeld.Verbfeld)	Gefundene semantische Beziehungen mit Hilfe des Verfahrens: Abstraktion des Verbs durch Recherche aus dem Wörterbuch (wissen.de) Durch die in Klammer gesetzte Ziffer (Ziffer) wird auf einen Kommentar zur Tabelle hingewiesen
... Ruhe und Bewegung	*laufen, fahren, rennen, gehen, stehen, ruhen, fliegen*	*(fahren, fliegen) → (fort)bewegen, lenken, reisen, fahren* *(rennen, laufen) → laufen* *(laufen, gehen) → gehen, entfernen,,fortbewegen* *(gehen,,rennen) → bewegen, gehen* *(stehen, gehen)→ kleiden*(1)
... Gebens und Nehmens	*schenken (7.1), verkaufen (7.1), kaufen (7.1), erben, erhalten (1.3)*	*(verkaufen, schenken) → geben*
... Gefühlsäußerungen	*lachen, weinen, trauern, erfreuen, freuen, seufzen*	*keine Übereinstimmungen gefunden*
... mentalen Prozesse	*denken (denken an 4.11) erfahren, vorstellen, meinen, wahrnehmen, vermuten, lesen, rechnen*	*(denken, vermuten) → annehmen, glauben*
... Sinneswahrnehmungen	*hören, sehen, riechen, tasten, schmecken, spüren*	*(hören, sehen, riechen, schmecken) → wahrnehmen* *(tasten, spüren) → suchen*
... Arbeitens	*arbeiten, backen, kochen, putzen, tragen (tragen, Rechnung 4.10), schrauben, feilen*	*(putzen, kochen) → säubern*(2)
... menschlichen Äußerungen	*rufen (ins Leben rufen 1.3), singen, sprechen (über 6.3), sagen (6.1), winken, zwinkern,*	*(rufen, singen) → ertönen* *(sprechen, rufen) → sprechen* *(sagen, rufen) → sagen*
... Bezeichnens	*nennen, heißen, bezeichnen, definieren*	*(heißen, bezeichnen) → bezeichnen* *(nennen, heißen) → nennen, sagen*
... Wohnens und Kleidens	*zudecken, einrichten, anziehen, wohnen*	*keine Übereinstimmungen gefunden*

Tabelle 7: Semantisches In-Beziehung-Setzen von Verben

(1) Die Lesarten für die Verben *stehen* und *gehen,* die das Hyperonym *kleiden* aufweisen, sehen folgendermaßen aus:

{...; II.1.*stehen*; etwas sieht gut an jmdm. aus, etwas *kleidet* jmdn.; das Kleid *steht* dir gut }

{...; I d; *gehen*; sich *kleiden*; sie *geht* gern in Rot }

Die Behauptung einer semantischen Übereinstimmung anhand der Feststellung, dass in diesem Fall beide Lesarten ein identisches Hyperonym haben, ist nicht richtig, da weitere Angaben zur Bedeutungserklärung zeigen, dass es sich hier um Redewendungen[86] oder Redensarten[87] handelt, die die Verben auf unterschiedliche Weise abstrahieren.

(2) Die festgestellte semantische Übereinstimmung zu *putzen, kochen → säubern w*ird in der Bedeutungserklärung von *kochen* sichtbar: Die Lesart, die das Verb *säubern* enthält, sieht folgendermaßen aus {...; I.2; *mit siedendem Wasser säubern;* } - also z. B. Wäsche kochen.

Mit Hilfe des Verfahrens „Zuordnung eines Hyperonyms zum Verb durch Recherche aus dem Wörterbuch (wissen.de)“ (siehe Kapitel 11.1) konnte festgestellt werden, dass einige Verben, die der gleichen Klassifikation angehören, vollständig mit einem Hyperonym in Verbindung gesetzt werden können. Z.B. Verben der Sinneswahrnehmung werden innerhalb der Bedeutungen mit dem Verb w*ahrnehmen* beschrieben.

Es zeigt sich jedoch auch, dass es Klassifikationen gibt, zu denen nur wenige oder keine Verben in Beziehung setzbar sind, wie die „Verben der Gefühlsäußerungen“ und „Verben der mentalen Prozesse“. Innerhalb der Verben der mentalen Prozesse können nur die Verben *denken* und *vermuten* mit *(an)nehmen, glauben* in Verbindung gesetzt werden.

Dagegen kommt es vor, dass Verben abstrahiert werden können, die keine semantische Übereinstimmung haben wie im Fall (1) oder eine semantische Übereinstimmung festgestellt wird wie im Fall (2), die aber abhängig vom verwendeten Objekt ist.

Der Satz *ich koche Wäsche* hat eine semantische Übereinstimmung mit *ich säubere Wäsche*. Diese Übereinstimmung gilt nicht, wenn z. B. *Wäsche* mit *Suppe* ausgetauscht wird.

Ebenso kann nicht die Aussage gemacht werden, dass Verben, die mit Hilfe dieses Verfahrens semantisch in Beziehung gesetzt werden, der gleichen Klassifikation angehören. Z.B. haben die Verben *hören* (Verben der Sinneswahrnehmung) und *rufen* (Verben menschlicher Äußerungen) die gemeinsamen Hyperonyme *sagen* und *nennen.*

Nur wenige Verben konnten innerhalb des Valenzwörterbuches „Verben in Feldern“ wiedergefunden werden. Dies ist damit begründbar, dass in dieser Ausgabe des

86 Redewendung: Verbindung von mehreren Wörtern mit fester Wortverbindung, die eine Einheit bilden und deren Gesamtbedeutung nicht direkt aus der Bedeutung der Einzelelemente abgeleitet werden kann.

87 Redensarten: feststehende sprachliche Wendungen, die in ihrem Kontext eingebunden und meist von symbolischer Bedeutung sind.

Valenzwörterbuches nicht alle sinnvoll existierenden Makrofelder enthalten sind, wie z. B. das Makrofeld „Verben der Ruhe und Bewegung“. In der Einleitung des Valenzwörterbuches ist dazu folgende Begründung zu finden:
„Aus arbeitstechnischen Gründen musste diese Zahl auf etwa die Hälfte reduziert werden. Um den Charakter eines Modellwörterbuchs zu wahren, wurde entschieden, für einen gewählten Begriff weitgehend alle verbalen Ausdrucksmöglichkeiten zu erfassen. Deshalb wurden die tentativ erstellten Felder nicht ´ausgedünnt´, vielmehr wurde auf einen Teil der erfassten Bereiche ganz verzichtet. Zu diesem gehörten u. a. die Verben, mit denen physikalische Handlungen ausgedrückt werden, ein größerer Teil der Verben des sprachlichen Ausdrucks und die Verben aus dem studentischen Bereich.“ [88]

Verben der physikalischen Handlungen sind mit der Klassifikation „Verben der Ruhe und Bewegung“ semantisch übereinstimmend, und somit ist diese Klassifikation mit dem vorliegenden Nachschlagewerk abdeckbar. Am ausgeprägtesten sind die Zuordnungen der Klassifikation der Verben des Gebens und Nehmens:
Die Verben *schenken (7.1), verkaufen (7.1), kaufen (7.1)* sind dem Verbfeld „Verben des Besitzes und Besitzwechsels“ zugeordnet.
Das Verb *erhalten* ist jedoch in einem anderen Verbfeld, und zwar in „Kausative Verben der allgemeinen Existenz“ wiederzufinden [Verbfeld (1.3)].

11.5 Musterübereinstimmungen der Nominal- und Präpositionalphrasen

Bei dem In-Beziehung-Setzen von Verben kann die semantische Übereinstimmung auch dadurch überprüft werden, dass eine Musterübereinstimmung innerhalb der Phrasen eines Satzes vorzufinden ist.

Wie schon in Kapitel 4.2 „Präpositionalphrasen“ dargestellt, spiegeln nach der Theorie von Fillmore die Kasi der Nominalphrasen und auch der Modus einer Präposition (z. B. hat die Präposition *mit* den Modus modal, instrumental) eine semantische Aussage wieder.

Innerhalb der Wort-Analyse werden die möglichen Kasi des Substantivs, des Artikels und des Possessiv- oder Demonstrativpronomens und die Modi der Präposition erkannt.

11.6 Experimentelle Auswertung der Musterübereinstimmungen

Im folgenden sollen innerhalb der semantischen Übereinstimmungen die Satzphrasen des eingegebenen Satzes und die Bedeutungen der Verben innerhalb des Satzes exemplarisch analysiert werden.

88 Schumacher: „Verben in Feldern“, S. 8-9

Als Beispiel wird der Satz *Ich fahre [mit (dem Auto)] [in (die Stadt)]* in das System (siehe „Realisierung des semantisch In-Beziehung-Setzens von Verben“) eingegeben.

Die Wort-Analyse generiert folgende Ausgabe:

Wort	Wortart	Attribut-Werte-Paare
Ich	*Personalpronomen*	*Kasus: "Nom.", Person: "1. Pers.", Numerus: "Sing.",*
fahre	*Verb*	*1.Person: "1. Pers. Sing./Imperativ Sing.", Tempus: "Praesens",*
mit	*Praeposition*	*Modus: "modal, instrumental,",*
dem	*Artikel*	*Kasus: "Dat./Dat.", Genus: "Mask./Neut.", Numerus: "Sing./Sing.", Art:"bestimmt",*
Auto	*Substantiv*	*Genus: "Neut.", Kasus: "Nom./Dat./Akk.", Numerus: "Sing./Sing./Sing.",*
in	*Praeposition*	*Modus: "lokal, temporal,",*
die	*Artikel*	*Kasus: "Nom./Akk./Nom./Akk.", Genus: "Fem./Fem./*/*/", Numerus: "Sing./Sing./Plu./Plu.", Art: "bestimmt",*
Stadt	*Substantiv*	*Genus: "Feminin", Kasus: "Nom./Akk./Dat./Gen.", Numerus: "Sing./Sing./Sing./Sing.",*
.	*Satzzeichen*	*Zeichen: "Punkt",*

Tabelle 8: Ergebnis einer Wort-Analyse

Die Phrasen-Analyse erkennt folgende Phrasen in dem Satz (Notation: Nominalphrasen stehen in runden Klammern () und Präpositionalphrasen in eckigen Klammern []):

Ich fahre [mit (dem Auto)] [in (die Stadt)].

Um an den Kasus der Nominalphrasen zu gelangen, wird die Schnittmenge der möglichen Kasi, Genus und Numeri des Artikels (wenn vorhanden), des Substantivs und der möglichen Kasi, die die entsprechende Präposition zulässt, gebildet.
Z. B. ist *mit* eine Präposition mit Dativ und *in* eine Präposition mit Dativ oder Akkusativ

Die möglichen Tripel aus Kasus, Genus und Numerus des Artikel „dem“ sind:
{Dat., Sing., Mask.} und {Dat., Sing., Neut.}.

Das Substantiv *Auto* ist vom Genus neutrum, entsprechend sind auch alle Tripel mit diesem Genus zu versehen: {Nom., Sing., Neut.}, {Akk., Sing., Neut.} und {Dat., Sing., Neut.}
Die Schnittmenge der Tripel des Artikels und des Substantivs sind: {Dat., Sing., Neut.}
Da die Präposition *mit* den Dativ einleitet, bleibtdas Ergebnis bei der o. g. Schnittmenge.

Auf gleiche Weise wird auch mit der Präpositionalphrase *in die Stadt* verfahren. Die in diesem Satz vorkommenden Nominal- und Präpositionalphrasen sind folgendermaßen zu parameterisieren:

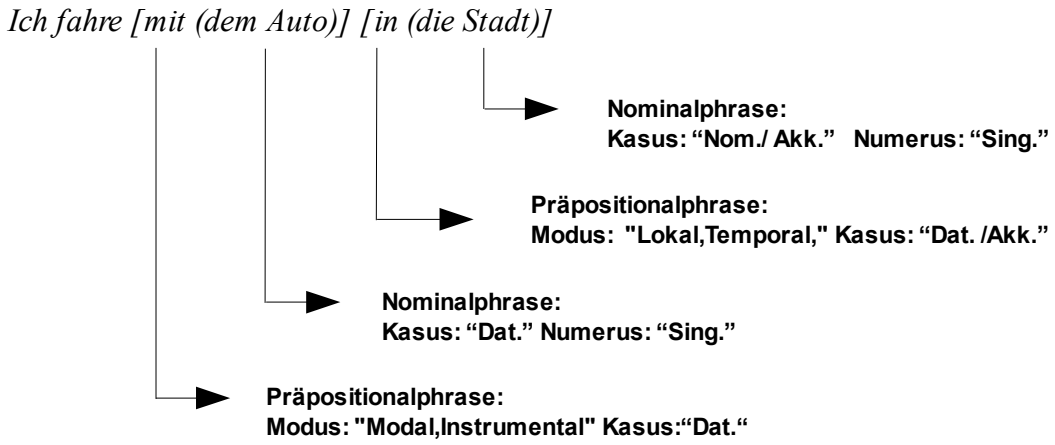


Abbildung 11: Phrasen innerhalb des deutschen Satzes "Ich fahre mit dem Auto in die Stadt"

Im Wörterbuchzugriff wissen.de ist folgende Lesart wiederzufinden: {..., I.1., *sich auf Rädern oder durch Triebkraft fortbewegen, ...* }, deren Phrasen folgendermaßen parameterisiert werden:

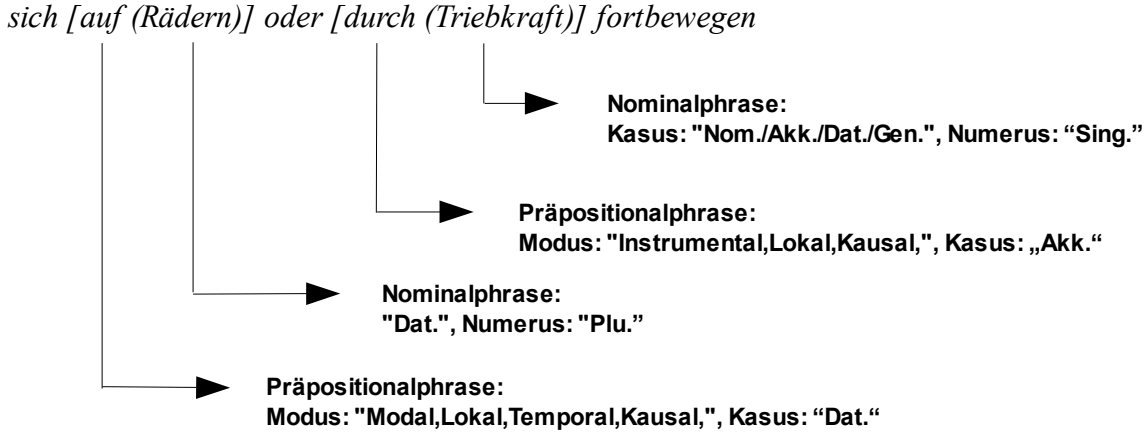


Abbildung 12: Phrasen innerhalb des deutschen Satzes "sich auf Rädern oder durch Triebkraft fortbewegen“

Da die Phrasen *auf Rädern* und *durch Triebkraft* in der Bedeutungserklärung (Lesart I.1. von *fahren*) semantisch die Hilfsmittel zu dem Verb *fahren* beschreiben, sollten diese mit der eingegebenen Phrasen *mit dem Auto* in Beziehung gesetzt werden können.

Übereinstimmungen ergeben sich innerhalb beider Präpositionalphrasen *auf Rädern* und *mit dem Auto"* mit dem Modus „Modal" und dem Kasus „Dativ".

Eine Übereinstimmung der Kasi *mit dem Auto* und *durch Triebkraft* ist nicht zu finden. Lediglich sind diese beiden Phrasen mit dem Modus „Instrumental" parameterisiert.

Jedoch existiert eine Übereinstimmung der Phrasen *durch Triebkraft* und *in die Stadt:*

Präpositionalphrase aus Bedeutungserklärung	**Präpositionalphrase aus Beispielsatz**	**übereinstimmende Modi und Kasi**
durch Triebkraft	*in die Stadt*	Modus: Lokal Kasus : Akkusativ

Tabelle 9: Übereinstimmende Modi und Kasi der Präpositionalphrasen „durch Triebkraft" und „in die Stadt"

Diese Beziehungen sind jedoch nicht erwünscht, da aus dem Verständnis dieses Satzes zu entnehmen ist, dass *durch Triebkraft* ein Hilfsmittel ausdrückt und *in die Stadt* eine Ortsangabe.

Das ist damit begründet, dass Präpositionen innerhalb der deutschen Sprache für mehrere Modi eingesetzt werden können:

Z.B. *durch ein Fahrzeug* --> Aussage instrumental
durch das Rohr --> Aussage lokal

Somit bestätigt sich die Aussage von Fillmore, dass es nicht möglich ist, alleine durch die Übereinstimmung der Oberflächen-Kasi, semantische Übereinstimmungen festzustellen.

12 Hyperbolische Bäume

Eine Methode Graphen (auch OEM-Graphen) visuell darzustellen und zwischen Knoten und Kanten zu navigieren, entwickelte die Firma Inxight[89] : die sog. hyperbolischen Bäume. Mit Hilfe dieser wird ein Graph (ein Baum mit Quer-Referenzen) auf eine Kugel gespannt dargestellt. Knoten (Objekte) können fokussiert werden, indem der Anwender diese mit Hilfe von Anklicken und Schieben der Maus (dragging) in die Mitte des Bildschirm-Fensters zieht.[90] Die umliegenden Objekte liegen darauf kreisförmig um das fokussierte Objekt.

Dieses Verfahren ist von der Firma Inxight patentiert worden. D. h. in Ländern, in denen Software patentierbar ist, dürfen Anwendungen für Visualisierung für hyperbolische Bäume nicht ohne Genehmigung durchgeführt werden. Ausgenommen davon ist Europa, wo auf Software keine Patentrechte gelten.[91]
Die Firma Inxight vertreibt für diese Anwender selber die Software StarTree[92].

Eine weitere, auf Basis von Open Source Lizenzierung zur Verfügung stehende Software ist der Treebolic Browser[93].
Der Treebolic Browser wird innerhalb dieses Projektes für die Darstellung eines wort- und phrasen-analysierten Satzes und des Verbfeld-Graphen eingesetzt. Diese liegen anhand von wohlgeformten semistrukturierten Daten vor und sind an das OEM-Modell angelehnt.

12.1 Darstellung von Graphen mit dem Treebolic Browser

Die Darstellung eines OEM-Modells mit dem Treebolic Browser ist jedoch mit folgenden Einschränkungen verbunden:
Lediglich die Knoten können beschriftet und zusätzlich mit einer Graphik (einem Symbol) visualisiert werden, jedoch nicht die Kanten. Somit sind wie bei der Notation XML und ssd nicht die Referenzen mit einem Attribut belegt, wie bei OEM-Modellen, sondern die Elemente enthalten einen Namen.
Dieser Name wird mit Hilfe der anzeigbaren Graphiken für einen Knoten repräsentiert. Da es nur eine begrenzte Anzahl von Typen gibt, die durch die Element-Namen (wie Wortarten, Phrasentypen, Wurzel, Satz, Bedeutungen etc.) ausgedrückt werden, existiert für jeden Knotentyp eine Grafikdatei (.gif-Format) im Verzeichnis `TreeBolic/images/` mit der entsprechenden Beschriftung.

89 Insight http://www.inxight.com (2.2007)
90 Vgl. Insight: „Insight StarTree“, http://www.inxight.com/products/sdks/st/ (2.2007)
91 Vgl. Hyperbolic [Patentierung von hyperbolischen Bäumen], http://treebolic.sourceforge.net/warning.htm (2.2007)
92 Insight: „Insight StarTree“, http://www.inxight.com/products/sdks/st (2.2007)
93 Treebolic, http://sourceforge.net/projects/treebolic (2.2007)

Für die Erhöhung der Übersichtlichkeit werden die Blätter (atomare Objekte, innerhalb der Notation ssd oder XML auch die Attribut-Werte-Paare) nicht direkt in die hyperbolischen Bäume übernommen, sondern die Attribut-Werte-Tupels werden aneinandergereiht und dem (im OEM-Graphen übergeordneten) Knoten als Beschriftung (`Label`) angefügt.

Eine Ausgabedatei in XML-Format für die Darstellung eines Graphen für den Treebolic Browser ist in Anhang D.2: „XML-Datei für Treebolic Browser“ zu finden. Dieser ist der DTD unterworfen, die im Anhang D.1: „DTD des Treebolic Browsers“ zu finden ist.

12.2 Generierung eines Verbfeld-Graphen und Visualisierung mit dem Treebolic Browser

Im folgenden wird der Satz *Ich fahre mit dem Auto in die Stadt* in das Eingabefeld innerhalb der Oberfläche des SentenceView eingegeben und der Anwendungsfall „Verbfeldgraph generieren und erweitern“ aufgerufen (siehe Kapitel 6.4 „Die Anwendung SentenceView“).

Darauf wird der Satz *Ich fliege mit dem Flugzeug über die Berge* eingegeben und obiger Anwendungsfall erneut aufgerufen.

Der Treebolic Browser wird geöffnet und die Datei `VerbFieldGraphReduced.xml` im Verzeichnis `/Treebolic` geladen.

Nun wird die Darstellung des hyperbolischen Baumes aktiviert und mit Hilfe des Mauspfeils bestimmte Bereiche fokussiert, um die eingegebenen Sätze mit Bedeutungserklärungen mit dem Hyperonym *fortbewegen* zu betrachten, die durch eine Quer-Referenz verbunden sind. (Eine ScreenShot der Quer-Referenz [blaue Linien innerhalb des hyperbolischen Baumes] soll hier aus Übersichtlichkeitsgründen nicht dargestellt werden.)

In Abbildung 13 ist der phrasen-analysierte Satz *Ich fahre mit dem Auto in die Stadt* innerhalb des Treebolic Browsers zu sehen.

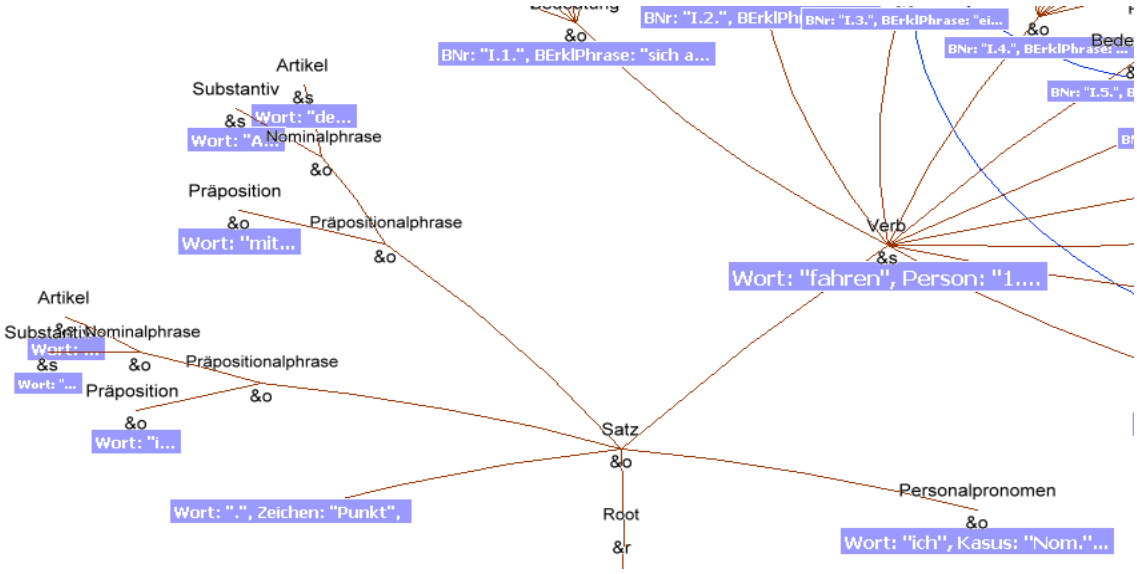


Abbildung 13: Screenshot aus Treebolic des Satzes "Ich fahre mit dem Auto in die Stadt“

In der folgenden Abbildung 14 ist die Bedeutungserklärung (Bedeutung) *sich auf Rädern oder durch Triebkraft fortbewegen* zu sehen, die dem Verb *fahren* angefügt ist:

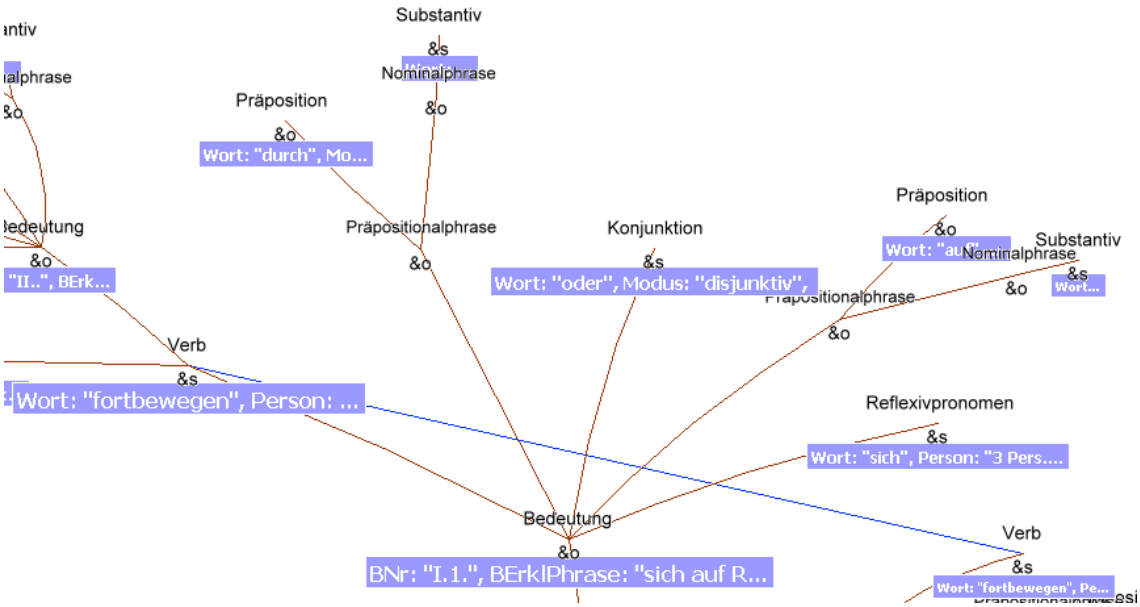


Abbildung 14: Screenshot aus Treebolic der Bedeutungserklärung "sich auf Rädern oder durch Triebkraft fortbewegen"

Ebenso kann der Satz *Ich fliege mit dem Flugzeug über die Berge* visualisiert werden:

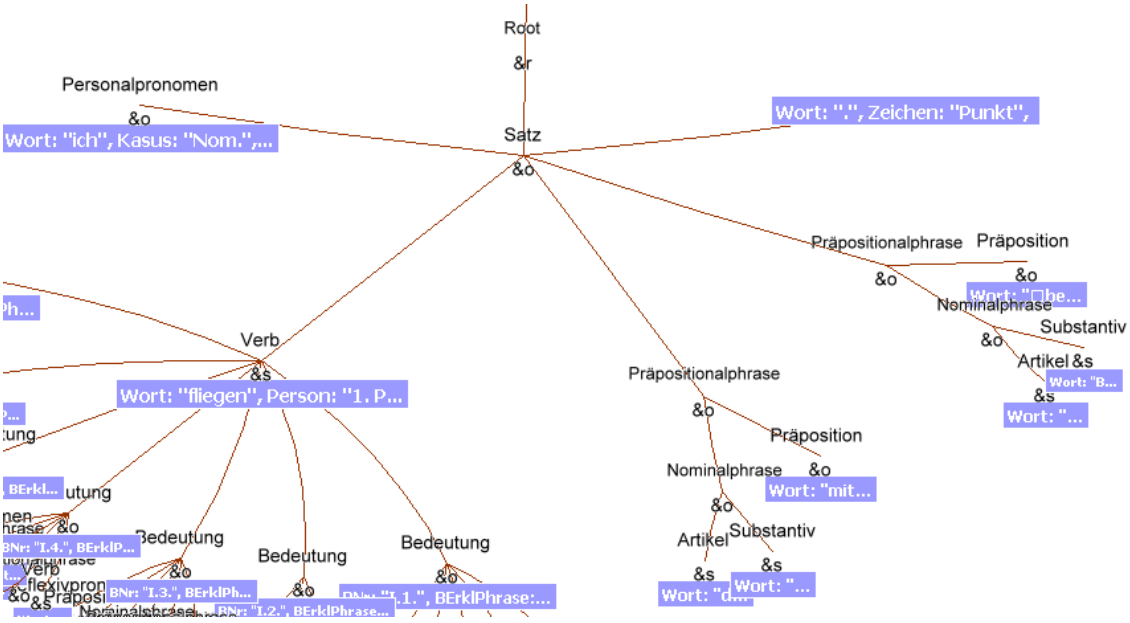


Abbildung 15: Screenshot aus Treebolic des Satzes "Ich fliege mit dem Flugzeug über die Berge"

Die Bedeutungserklärung des Verbs *fliegen = sich mit einem Luftfahrzeug fortbewegen, mit dem Flugzeug reisen* sieht folgendermaßen aus:

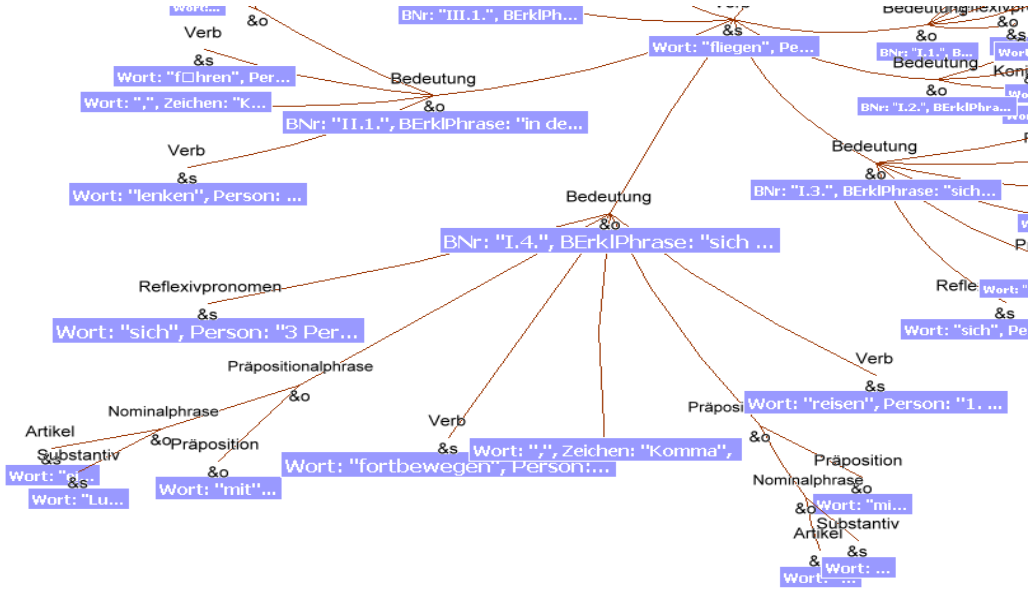


Abbildung 16: Screenshot aus Treebolic der Bedeutungserklärung "sich mit einem Luftfahrzeug fortbewegen, mit dem Flugzeug reisen"

12.3 Auswertung des visualisierten Graphen

In Abbildung 5 wurde ein Graph dargestellt, der einen Satz beschreibt; dem Verb ist ein Element mit der Referenz „Bedeutung“ untergeordnet, dem wiederum Elemente für die Beschreibung von Wörtern und Phrasen der Bedeutungserklärung angehängt sind.

Innerhalb des generierten Graphen, der zwei eingegebene Sätze mit ihren Bedeutungen beschreibt, sind nun zwei Elemente mit der Referenz „Satz“ der Wurzel untergeordnet.

Die Elemente mit der Referenz „Bedeutung“ der eingegebenen Sätze, die dem Hyperonym *fortbewegen* übergeordnet sind, weisen beide Präpositionalphrasen auf, die im folgenden mit dem Verfahren der Musterübereinstimmung (siehe auch Kapitel 11.6 „Experimentelle Auswertung der Musterübereinstimmungen“) untersucht werden.

Die Präpositionalphrasen der analysierten Bedeutungserklärung des ersten eingegebenen Satzes lauten *„sich auf Rädern oder durch Triebkraft fortbewegen“*.

Präpositionalphrasen	**Modi und Kasi**
auf Rädern	Modus: Temporal, Lokal, Kausal, Modal, Kasus: Dativ
durch Triebkraft	Modus: Instrumental, Lokal, Kausal, Kasus: Akkusativ

Tabelle 10: Modi und Kasi der Präpositionalphrasen „auf Rädern“ und „durch Triebkraft“

Im folgenden ist die Präpositionalphrase der Bedeutungserklärung des zweiten Satzes *sich mit einem Luftfahrzeug fortbewegen, ...* beschrieben:

Präpositionalphrase	**Modi und Kasi**
mit einem Luftfahrzeug	Modus: Modal, Instrumental, Kasus: Dativ

Tabelle 11: Modi und Kasi der Präpositionalphrase „mit einem Luftfahrzeug“

Semantisch zeigt sich, dass die Präpositionalphrasen *auf Rädern* und *durch Triebkraft* mit der Präpositionalphrase *mit einem Luftfahrzeug* in Beziehung zu setzen sind, da alle drei Nominalphrasen ein Hilfsmittel zur Fortbewegung ausdrücken.

Übereinstimmungen mit dem Phrasen des ersten eingegebenen Satzes und den Phrasen des zweiten ergeben sich wie folgt:

Präpositionalphrase der analysierten Bedeutungserklärung des ersten eingegebenen Satzes	**Präpositionalphrase** der analysierten Bedeutungserklärung des zweiten eingegebenen Satzes	**übereinstimmende Modi und Kasi**
auf Rädern	*mit einem Luftfahrzeug*	Modus: Modal Kasus : Dativ
durch Triebkraft		Modus: Instrumental Kasus : (keine Übereinstimmung)

Tabelle 12: Übereinstimmende Modi und Kasi der Präpositionalphrasen „auf Rädern“ und „durch Triebkraft“ mit der Präpositionalphrase “mit einem Luftfahrzeug“

Somit ist dieses Ergebnis, dass durch das In-Beziehung-Setzen der Modi und Kasi der Phrasen semantisch Übereinstimmendes nicht gefunden werden kann, identisch mit dem Ergebnis in Kapitel 11.6 „Experimentelle Auswertung der Musterübereinstimmungen“, wo die Präpositionalphrasen des ersten eingegebenen Satzes mit der Präpositionalphrase *mit dem Auto* in Beziehung gesetzt werden.

Resümee

Abschließend möchte ich den Entwicklungsstand dieser Masterarbeit charakterisieren und wiedergeben, was ich damit an Erkenntnissen gewonnen und gelernt habe. Es sollen auch Gedanken und Anregungen zur Weiterentwicklung zur Sprache kommen und zu anderen Projekten Bezug genommen werden.

12.4 Der Einsatz von IPEE innerhalb der Wort- und Phrasen-Analyse

Dieses Unterkapitel ist aus Gründen des Copyrights, die auf der Softwarelösung IPEE liegen, nicht in der veröffentlichten Masterarbeit enthalten.

Semantisch orientierte Verarbeitung

Mit dieser Masterarbeit habe ich einen Prototypen konstruiert, der einen interessanten Teilaspekt der semantischen Verarbeitung demonstriert, auch wenn es sich gezeigt hat, dass eine erwartete semantische Beziehung allein mit dem Verfahren „Zuordnung eines Hyperonyms zum Verb durch Recherche aus dem Wörterbuch (wissen.de)“ nur tendenzweise zu einem erwünschten Ergebnis führt.

Dies soll anhand von einigen statistischen Aussagen gezeigt werden: Innerhalb der an das SKV angelehnten Tabelle 7, die insgesamt 53 zugrunde liegende Verben enthält, existieren davon 26 Verben, zu denen eine semantische Beziehung mit Hilfe eine Hyperonyms zu einem anderen geführt werden könnte. Das ergibt eine Güte von 0,491 (annähernd jedes zweite Verb).

Innerhalb des Experiments wurde eine falsche, semantische Übereinstimmung gefunden, und zwar siehe (1) in Tabelle 4 und eine, bei der eine semantische Übereinstimmung von dem Objekt des Satzes abhängig ist, siehe (2) in Tabelle 7. Beides ergibt einen Quotient von 0,02.

Ideal ist, wenn alle Verben einer semantischen Klassifikation einem einzigen Hyperonym zugeordnet werden können. Eine solche Klassifikation kommt jedoch nicht vor. Die Klassifikation der „Sinneswahrnehmungen“, bei der vier von sechs Verben dem Hyperonym *wahrnehmen* zugeordnet werden können, nähert sich diesem Ideal. Bei allen weiteren Klassifikationen können maximal drei Verben einem Hyperonym zugeordnet werden.

Obwohl nur ein Teilziel erreicht wurde, hat mich die Beschäftigung mit der semantisch orientierten Verarbeitung dieser gesamten Problematik näher gebracht, und die Konzeption meiner Masterarbeit versteht sich als Grundlage für weitere Experimente und Entwicklungen.

Abzuwarten ist, ob zu dem Valenzwörterbuch „Verben in Feldern“ weitere Ausgaben folgen mit einer vervollständigten Dokumentation von Verben der deutschen Sprache. Wie das Experiment aus Kapitel 11 „Semantisch orientierte Verarbeitung“ zeigt, sind nur wenige (9 von 53) Verben wiederzufinden; von diesen neun Verben sind einige im Valenzwörterbuch auch nur unter Anwendung bestimmter Präpositionen wie *sprechen über (6.3)* oder Phrasen wie *(ins Leben rufen 1.3)* wiederzufinden.

Es erhebt sich die Frage: Gibt es eine Möglichkeit zur Optimierung, indem beide in Kapitel 11 „Semantisch orientierte Verarbeitung“ vorgestellten Verfahren zum semantisch In-Beziehung-Setzen von Verben sinnvoll kombiniert angewandt werden?

Eine grundlegende Idee zur Weiterentwicklung ergibt sich aus der Fokussierung der semantisch orientierten Verarbeitung auf Verben. Z.B. könnten in ähnlicher Weise auch Substantive abstrahiert und semantische Übereinstimmungen erkannt werden.

Das Internet-Wörterlexikon Wortschatz Universität Leipzig[94] bietet bei Nachfrage der Substantive unter anderem Unterbegriffe und Oberbegriffe (*Verkehrsmittel* ist z. B. Oberbegriff von *Auto*) unter der Spalte „ist ein(e)“ an.

Auch wäre die Betrachtung von Teil/Ganzes-Beziehungen von Substantiven relevant. Damit könnten auch die Substantive innerhalb der Nominal- und Präpositional-Phrasen zur Entscheidung einer semantischen Beziehung herangezogen werden.

Für ein semantisch orientiertes, verarbeitendes System, das evtl. eine Suchmaschine für Dokumente im Internet einsetzt, bestehen weitere Anforderungen, wie z. B. das Erkennen von Redewendungen. „*Die Maschine ist im Gang*“ ist semantisch gleichzusetzen mit „*die Maschine läuft“*. Oder die semantische Abstraktion einer Teil/Ganzes-Beziehung: „*Der Motor des Autos ist kaputt“* hat eine semantisch ähnliche Aussage wie der Satz *„Das Auto ist kaputt“*.

Somit existiert eine ganzes Kalkül von möglichen weiteren Operationen, die zur semantischen Abstraktion eines Satzes von Natursprachen dienen können.

Möglichkeiten zur weiteren Entwicklung bietet auch das „Case Grammar Model“ von Fillmore. Damit könnten Analyseverfahren zur semantisch orientierten Verarbeitung noch weiter ausgebaut werden. Mit Hilfe einer der Wort- und Phrasen-Analyse übergeordneten Satz-Analyse können Aktiv- oder Passiv- Konstruktionen von Sätzen der deutschen Sprache erkannt werden. Dies gibt Auskunft, ob ein Substantiv im Nominativ dem Tiefenkasus

94 Wortschatz Universität Leipzig, http://www.wortschatz.uni-leipzig.de (2.2007)

Agentiv (Auslöser einer Aktion [95]) entspricht oder instrumental (in einem Zustand oder einer Handlung) involviert ist [96].

Einschränkungen

Der englische Philosoph John Locke ist einer der Hauptvertreter des Empirismus, einer Philosophie, die die Erfahrung zu ihrer Grundlage macht.
Seine Erkenntnistheorie sagt aus, dass durch Vorgänge wie *Abstrahieren, Vergleichen, Verbinden, Trennen* durch äußere Sinne oder selbst wahrgenommene Ideen weitere komplexere Ideen entstehen. [97]

Das in dieser Masterarbeit vorgestellte Verfahren basiert in übertragener Weise auf Vorgänge des empirischen Wahrnehmens: Eingabe eines Satzes, Recherche im Lexikon und Weiterverarbeiten (Finden einer semantischen Übereinstimmung).

Es kann nicht ausgeschlossen werden, dass - auch wenn dieser vorgestellte Prototyp zu einem leistungsfähigeren System weiter entwickelt wird - durch die Bildung des semantischen In-Beziehung-Setzens die Aussage verfälscht wird. Denn dabei muss abstrahiert werden. Das bedeutet, dass bestimmte Aspekte eines Elementes weggenommen werden, um dieses Element mit einem übergeordneten zu verbinden. Der gegenteilige Prozess, die Zuordnung eines übergeordneten zu einem weiteren Element, bedeutet wieder ein Hinzufügen von anderen Aspekten. Dies kann grundsätzlich zu einer semantischen Verfälschung führen. Somit kann keine Garantie auf Richtigkeit einer Zuordnung gegeben werden.

Für den praktischen Einsatz stellt sich die Frage, inwieweit der Benutzer in diesem Sinne ungenaue Ergebnisse anwenden kann.

Die Aufgabe der Entwicklung von semantisch orientierten, verarbeitenden Systemen liegt darin, die Genauigkeit zu erhöhen.

95 Vgl. Fillmore: „Fillmore's case grammar“, S. 24
96 Vgl. Fillmore: „Fillmore's case grammar“, S. 24
97 Vgl. Kunzmann, Burkard, Wiedmann: „Philosophie“, S. 119

Literaturverzeichnis

Abiteboul, Serge; Buneman, Peter; Suciu, Dan: „Date on the Web, From Relations to Semistructured Data on XML“, Morgan Kaufmann Publishers, San Francisco, Californien, 2000

Apache Software Foundation: „Apache HTTP Client“, 2001-2006, http://jakarta.apache.org/commons/httpclient (2.2007)

Beckstein Clemens; Sack, Harald; Peter, Heiko: “Semantic Web“, 2006, http://www.informatik.uni-jena.de/~hpeter/Lehre/Seminare/Semantic_Web_SS06/Seminarankuendigung/Seminarankuendigung.html (2.2007)

Boag, Scott et al. (Hg.): „XQuery 1.0: An XML Query Language“, 23.1.2007, http://www.w3.org/TR/xquery/ (2.2007)

Büchel, Gregor: „Semantische Klassifikation von Verben (SKV)“, 17.7.2006, Anhang B
Canoo Engineering: „Wörterbücher und Grammatik für Deutsch“, 2002-2007, http://www.canoo.net (2.2007)

Büchel, Gregor: „Strukturierung von Bedeutungserklärungen mit XMLTechnologien am Beispiel von Verben“, IKP-Arbeitsbericht NF 22, 08.2007, http://www.ifk.uni-bonn.de/forschung/abteilung-sprache-und-kommunikation/ikp-arbeitsberichte-neue-folge/ikpab-nf22.pdf (2.2008)

Büchel, Gregor: [Zusätzliche Verben für das Bedeutungswörterbuch], 2006, Anhang C.4
Clark, Andy: „CyberNeko HTML Parser“, 2002 - 2005, http://people.apache.org/~andyc/neko/doc/html (2.2007)

Clark, James; DeRose, Steve: „XML Path Language (Xpath) Version 1.0“, 16.11.1999, http://www.w3.org/TR/xpath (2.2007)

Dischert, Thorsten; Kiefel, Eduard: „Datenbankgestütztes, www-geführtes System zur Analyse präpositionaler Phrasen und zur Extraktion von Satzgliedern der deutschen Sprache auf UNIX-Servern“ 1998, Köln; Diplomarbeit FH Köln, Institut für Nachrichtentechnik Abgabe: 27.4.1998

Eisenberg, Peter et al. (Hg.): „Duden Band 4, Die Grammatik“, 7., völlig neu erarbeitete und erweiterte Auflage, Dudenverlag, Mannheim, 2005

Sandkühler, Hans Jörg: „Enzyklopädie Philosophie“, Meiner, Hamburg, 1999

Fillmore, Charles: „Fillmore's case grammar“, Julius Groos Verlag, Heidelberg, 1987

Frey, Andreas: „ICEE – Internet Car Extract Engine“, 2005, Köln Praxissemesterbericht FH Köln, Institut für Nachrichtentechnik. Abgabe: 29.8.2005

Greiner, Ulrike: „Semistrukturierte Daten“, 1999, Kapitel 1.1: http://dbs.uni-leipzig.de/html/seminararbeiten/semSS99/arbeit3/kapitel1.html, Kapitel 2.1.1 : „Datenmodelle und Schemata für semistrukturierte Daten“, http://dbs.uni-leipzig.de/html/seminararbeiten/semSS99/arbeit3/kapitel2.html, Kapitel :5.1.1: „Abfrage semistrukturierter Daten“ http://dbs.uni-leipzig.de/html/seminararbeiten/semSS99/arbeit3/kapitel5.html (2.2007)

Hage, Simon: „Der Web-Meister“, 22.11.2006, http://www.manager-magazin.de/it/artikel/0,2828,449911,00.html, „Computer werden halb menschlich“, http://www.manager-magazin.de/it/artikel/0,2828,449911-4,00.html (2.2007)

Herman, Ivan; Swick, Ralph; Brickly, Dan: „Resource Description Framework (RDF)“, 29.1.2007, http://www.w3.org/RDF/ (2.2007)

Hyperbolic [Patentierung von hyperbolischen Bäumen], o.A., o.A., http://treebolic.sourceforge.net/warning.htm (2.2007)

Insight: „Insight StarTree“, 2006, http://www.inxight.com/products/sdks/st (2.2007)

Insight: „Transforming Text into Actionable Information“, 2007, http://www.inxight.com (2.2007)

Johann, Bernd Michael: „Datenbanksystem zur Analyse von Satzgliedtypen der Deutschen Sprache“, 2006, Köln; Diplomarbeit FH Köln, Institut für Nachrichtentechnik. Abgabe: 29.6.2006

Kling, Bernd: „Tim Berners-Lee verkündet das Web 3.0“, 24.10.2006, http://de.theinquirer.net/2006/09/24/tim_bernerslee_verkuendet_das.html (2.2007)

Kunzmann, Peter; Burkard, Frank-Peter; Wiedmann, Frank: “Philosophie”, Deutscher Taschenbuch Verlag, 1991

L. McGuinness, Deborah, van Harmelen, Frank: „OWL Web Ontology Language“, 10.2.2004, http://www.w3.org/TR/owl-features/ (2.2007)

Lang, H.W.: “Grundlagen Graph“, 29.01.1998, http://www.inf.fh-flensburg.de/lang/algorithmen/grundlagen/graph.htm (10.1.2006)

Lang, H.W.: „Knoten und Kanten“, 29.01.1998, letzte Änderung:11.12.2005, http://www.inf.fh-flensburg.de/lang/algorithmen/grundlagen/graph.htm (2.2007)

Lausen, Georg: “Kapitel IV, Semi-strukturierte Daten: XML“, 2003, http://www.informatik.uni-freiburg.de/~dbis/lehre/db2-ss03/db2_030722.pdf (10.2005)

Locke, John: „Versuch über den menschlichen Verstand“, Erster Band, Verlag von Felix Meiner, Leipzig, 1913

Münz, Stefan: „SELFHTML: Version 8.1.1“, 24.11.2005, http://de.selfhtml.org, Absatz: “Wohlgeformtheit eines XML-Dokuments“ http://de.selfhtml.org/xml/regeln/begriffe.htm#wohlgeformt, Absatz: „Parameter Entities für komplexere DTDs“ http://de.selfhtml.org/xml/dtd/entities.htm#parameter_entities (2.2007)

Pemberton, Steven: „HyperText Markup Language (HTML)“, World Wide Web Consortium,1995-2006, http://www.w3.org/MarkUp (2.2007)

Quin, Liam: „Extensible Markup Language (XML)“, World Wide Web Consortium,1996 - 2003, letzte Änderung: 11.9.2006, http://www.w3.org/XML (2.2007)

Sandkühler, Hans Jörg: „Enzyklopädie Philosophie“, Meiner, Hamburg, 1999

Sauer, Hermann: „Relationale Datenbanken Theorie und Praxis“, Auflage 5. Addison-Wesley, München, 2002

Schumacher, Helmut: „Verben in Feldern, Valenzwörterbuch zur Syntax und Semantik deutscher Verben“, Schriften des Institutes für deutsche Sprache, Berlin, 1986

Sedgewick, Robert: „Algorithmen in C“, 1. Auflage, Addison-Wesley, München, 1992

Sun Microsystems: “java.util.regex Class Pattern“, 2004, http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html (2.2007)

Sun Microsystems: “JavaTM 2 Platform Standard Ed, 5.0“, 2004, http://java.sun.com/j2se/1.5.0/docs/api/index.html (2.2007)

Sun Microsystems: „JavaTM 2 Platform Standard Edition 5.0API Specification“, 2004, http://java.sun.com/j2se/1.5.0/docs/api/ (2.2007)
Treebolic, o.A., 2007, http://sourceforge.net/projects/treebolic/ (2.2007)

Treebolic: „Transforming Text into Actionable Information“, o.A., http://sourceforge.net/projects/treebolic

wissen.de: Schlagwort: „fahren“, o.A., http://www.wissen.de/wde/generator/wissen/services/suche/wbger/index.html?gerqry=fahren (2.2007)

wissen.de: Schlagwort: „fliegen“, o.A., http://www.wissen.de/wde/generator/wissen/services/suche/wbger/index.html?gerqry=fliegen (2.2007)

Wortschatz Universität Leipzig,1998 - 2006, http://www.wortschatz.uni-leipzig.de/ (2.2007)

Anhang A: Pattern-Extraktion

Dieses Kapitel ist aus Gründen des Copyrights nicht in der veröffentlichten Masterarbeit enthalten.

Anhang B: Semantische Klassifikation von Verben

Semantische Klassifikation von Verben (=:SKV) 17.7.2006 – Prof. Dr. phil. G. Büchel

1) V. der Ruhe und Bewegung	*(laufen, fahren, rennen, gehen, stehen, ruhen)*
2) V. des Gebens und Nehmens	*(schenken, verkaufen, kaufen, erben, erhalten)*
3.)$_{a)}$ V. der Gefühlsäußerungen	*(lachen, weinen, trauern, erfreuen, freuen, seufzen)*
4.)$_{a)}$ V. mentaler Prozesse	*(denken, erfahren, vorstellen, meinen, wahrnehmen, vermuten, lesen, rechnen, ...)*
$_{b)}$ V. der Sinneswahrnehmungen	*(hören, sehen, riechen, tasten, schmecken, spüren)*
5.) V. des Arbeitens	*(arbeiten, backen, kochen, putzen, tragen, schrauben, feilen)*
3.)$_{b)}$ V. menschlicher Äußerungen	*(rufen, singen, sprechen, räuspern, sagen, winken, zwinkern, [husten, schniefen, kotzen, zittern] ...) []~ eher krankhafte Äußerungen*
6.) V. des Bezeichnens	*(nennen, heißen, bezeichnen, definieren)*
7.) V. des Wohnens und Kleidens	*(zudecken, einrichten, anziehen, wohnen ,...)*

...

Anhang C: Bedeutungs-Wörterbuch

C.1: Erläuterung der Tag-Namen

7.2006 – Prof. Dr. phil. G. Büchel :

BedWB:	Bedeutungswörterbuch
WArtikel:	Wörterbuchartikel
SchlagW:	Schlagwort
ArtEin:	Artikeleintrag
SGramAn:	Grammatische Ausgabe zum Schlagwort
BfolgE:	Bedeutungsfolgenelement
BNr:	Bedeutungsnummer
BGramAn:	Grammatische Angabe zur Bedeutungsnummer
BErKl:	Bedeutungserklärung (= Verbalphrase, die auf einen Infinitiv endet)
BErWA:	Weitere Angaben zur Bedeutungserklärung (Beispiele o.ä.)
Sinnv:	Liste sinnverwandter Wörter innerhalb der Bedeutungserklärung zum Schlagwort
Zus:	Angaben von Zusammensetzungen bzw. Ableitungen

C.2: DTD-Datei BedWB/BedWB.DTD für Bedeutungs-Wörterbuch

```
<!ELEMENT BedWB (WArtikel+)>
<!-- EIn Woerterbuch hat mind. 1 Artikel -->
<!ELEMENT WArtikel (SchlagW,ArtEin*)>
<!ELEMENT SchlagW (#PCDATA)>
<!ELEMENT ArtEin (SGramAn?, BFolgE+) >
<!-- Pro Artikeleintag gibt es hoechstens 1 gram. An -->
<!-- gabe, die das Schlagwort betrifft. -->
<!ELEMENT SGRamAn (#PCDATA) >
<!ELEMENT BFolgE (BNr,DefNr,BGramAn,BErkl*,BErWA*,Sinnv*,Zus*)>


<!-- Pro Bedeutungsfolgeelement gibt es immer eine -->
<!-- Bedeutungserklaerung, aber nicht immer eine BNr. -->
<!-- Dieses gilt, wenn die Folge nur ein Element hat. -->
<!ELEMENT BNr (#PCDATA) >
<!ELEMENT BGramAn (#PCDATA) >


<!-- folgendes Entity Element ist von -->
<!-- Author: Andreas Frey -->
```

```
<!-- Matr: 11013600 -->
<!-- Datum 1.10.2006 -->


<!ENTITY % Satzelemente "( Nominalphrase | Praepositionalphrase |
                  Kardinalzahl | Ordnungszahl | Zahlenwort |
                  Praeposition | Konjunktion |
                  Personalpronomen | Possessivpronomen |
                  Reflexivpronomen | Demonstrativpronomen |
Interrogativpronomen | Indefinitpronomen |
                  Artikel | Hilfsverb | Verb | Adjektiv | Adverb |
Substantiv | Partizip |
                  Satzzeichen | Abkuerzung | Masszahl |
notmatched )* ">


<!-- Dieses gilt, wenn die Folge nur ein Elment hat. -->
<!ELEMENT BErkl (%Satzelemente;) >
<!ATTLIST BErkl
  BErklPhrase CDATA #IMPLIED
>


<!ELEMENT BErWA (#PCDATA) >
<!ELEMENT Sinnv (#PCDATA) >
<!ELEMENT ZUs (#PCDATA) >


<!-- folgendes Element bis Datei Ende sind von -->
<!-- Author: Andreas Frey -->
<!-- Matr: 11013600 -->
<!-- Datum 1.10.2006 -->

<!ELEMENT DefNr (#PCDATA) >

<!-- Satzstrukturelmente fuer Bedeutungserklaerung (Berkl) und
Attribute -->

<!ELEMENT Praepositionalphrase (Praeposition?,Nominalphrase?) >
<!ELEMENT Nominalphrase (%Satzelemente;) >


<!ELEMENT Kardinalzahl (#PCDATA)>
<!ATTLIST Kardinalzahl
  Wort CDATA #IMPLIED
  Wert CDATA #IMPLIED
>

<!ELEMENT Ordnungszahl (#PCDATA)>
<!ATTLIST Ordnungszahl
  Wort CDATA #IMPLIED
  Wert CDATA #IMPLIED
>
```

```
<!ELEMENT Zahlenwort (#PCDATA)>
<!ATTLIST Zahlenwort
  Wort CDATA #IMPLIED
  Wert CDATA #IMPLIED
>

<!ELEMENT Praeposition (#PCDATA)>
<!ATTLIST Praeposition
  Wort CDATA #IMPLIED
  Modus CDATA #IMPLIED
>

<!ELEMENT Konjunktion (#PCDATA)>
<!ATTLIST Konjunktion
  Wort CDATA #IMPLIED
  Modus CDATA #IMPLIED
>

<!ELEMENT Personalpronomen (#PCDATA)>
<!ATTLIST Personalpronomen
  Wort CDATA #IMPLIED
  Kasus CDATA #IMPLIED
  Person CDATA #IMPLIED
  Numerus CDATA #IMPLIED
  Genus CDATA #IMPLIED
>

<!ELEMENT Possessivpronomen (#PCDATA)>
<!ATTLIST Possessivpronomen
  Wort CDATA #IMPLIED
  Kasus CDATA #IMPLIED
  Person CDATA #IMPLIED
  Numerus CDATA #IMPLIED
  Genus CDATA #IMPLIED
>

<!ELEMENT Reflexivpronomen (#PCDATA)>
<!ATTLIST Reflexivpronomen
  Wort CDATA #IMPLIED
  Kasus CDATA #IMPLIED
  Person CDATA #IMPLIED
  Numerus CDATA #IMPLIED
>

<!ELEMENT Demonstrativpronomen (#PCDATA)>
<!ATTLIST Demonstrativpronomen
  Wort CDATA #IMPLIED
  Kasus CDATA #IMPLIED
  Genus CDATA #IMPLIED
  Person CDATA #IMPLIED
>

<!ELEMENT Interrogativpronomen (#PCDATA)>
<!ATTLIST Interrogativpronomen
```

```
  Wort CDATA #IMPLIED
  Kasus CDATA #IMPLIED
  Genus CDATA #IMPLIED
  Person CDATA #IMPLIED
>

<!ELEMENT Indefinitpronomen (#PCDATA)>
<!ATTLIST Indefinitpronomen
  Wort CDATA #IMPLIED
  Kasus CDATA #IMPLIED
  Kuerzel CDATA #IMPLIED
>

<!ELEMENT Artikel (#PCDATA)>
<!ATTLIST Artikel
  Wort CDATA #IMPLIED
  Kasus CDATA #IMPLIED
  Genus CDATA #IMPLIED
  Numerus CDATA #IMPLIED
  Art CDATA #IMPLIED
>

<!ELEMENT Adverb (#PCDATA)>
<!ATTLIST Adverb
  Wort CDATA #IMPLIED
  Modus CDATA #IMPLIED
>

<!ELEMENT Hilfsverb (#PCDATA)>
<!ATTLIST Hilfsverb
  Wort CDATA #IMPLIED
  Grundform CDATA #IMPLIED
>

<!ELEMENT Verb (#PCDATA)>
<!ATTLIST Verb
  Wort CDATA #IMPLIED
  Person CDATA #IMPLIED
  Tempus CDATA #IMPLIED
  Praefix CDATA #IMPLIED
>

<!ELEMENT Adjektiv (#PCDATA)>
<!ATTLIST Adjektiv
  Wort CDATA #IMPLIED
  Steigerungsform CDATA #IMPLIED
>

<!ELEMENT Substantiv (#PCDATA)>
<!ATTLIST Substantiv
  Wort CDATA #IMPLIED
  Kasus CDATA #IMPLIED
  Numerus CDATA #IMPLIED
  Genus CDATA #IMPLIED
```

```
>

<!ELEMENT Partizip (#PCDATA)>
<!ATTLIST Partizip
  Wort CDATA #IMPLIED
  Steigerungsform CDATA #IMPLIED
>

<!ELEMENT Satzzeichen (#PCDATA)>
<!ATTLIST Satzzeichen
  Wort CDATA #IMPLIED
  Zeichen CDATA #IMPLIED
>

<!ELEMENT Abkuerzung (#PCDATA)>
<!ATTLIST Abkuerzung
  Wort CDATA #IMPLIED
  Kuerzel CDATA #IMPLIED
>

<!ELEMENT Masszahl (#PCDATA)>
<!ATTLIST Masszahl
  Wort CDATA #IMPLIED
  Kuerzel CDATA #IMPLIED
>

<!-- Tag fuer nicht erkanntes Wort -->
<!ELEMENT notmatched (#PCDATA)>
<!ATTLIST notmatched
  Wort CDATA #IMPLIED
>
```

C.3: Auschnitt aus /BedWB/BedWBWissen.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE BedWB SYSTEM "BedWB/BedWB.dtd">
<BedWB>

...

      <Wartikel>
            <SchlagW>bewegen</SchlagW>
            <ArtEin>
                  <BFolgE>
                        <BNr>I.1.</BNr>
                        <DefNr>0</DefNr>
                        <BGramAn>[V.1</BGramAn>
                        <BErkl BErklPhrase="die Lage, Stellung von etwas
verändern, von der Stelle rücken">
                              <Nominalphrase>
                                    <Artikel Wort="die"
Kasus="Nom./Akk./Nom./Akk." Genus="Fem./Fem./*/*/" Numerus="Sing./Sing./Plu./Plu."
Art="bestimmt"/>
                                    <Substantiv Wort="Lage" Genus="feminin"
Kasus="Nom./Akk./Dat./Gen." Numerus="Sing./Sing./Sing./Sing."/>
                              </Nominalphrase>
                              <Satzzeichen Wort="," Zeichen="Komma"/>
                              <Nominalphrase>
                                    <Substantiv Wort="Stellung"
Genus="feminin" Kasus="Nom./Akk./Dat./Gen." Numerus="Sing./Sing./Sing./Sing."/>
                              </Nominalphrase>
                              <Praeposition Wort="von"
Modus="Modal,Temporal,Lokal,Kausal,"/>
                              <Zahlenwort Wort="etwas" Wert="etwas"/>
                              <Verb Wort="verändern" Person="1. Pers. Plu./3.
Pers. Plu./Infinitiv" Tempus="Praesens" Praefix="no"/>
                              <Satzzeichen Wort="," Zeichen="Komma"/>
                              <Praepositionalphrase>
                                    <Praeposition Wort="von"
Modus="Modal,Temporal,Lokal,Kausal,"/>
                                    <Nominalphrase>
                                          <Artikel Wort="der"
Kasus="Nom./Gen./Dat./Gen." Genus="Mask./Fem./Fem./*/"
Numerus="Sing./Sing./Sing./Plu." Art="bestimmt"/>
                                          <Substantiv Wort="Stelle"
Genus="feminin" Kasus="Nom./Akk./Dat./Gen." Numerus="Sing./Sing./Sing./Sing."/>
                                    </Nominalphrase>
                              </Praepositionalphrase>
                              <notmatched Wort="rücken"/>
                        </BErkl>
                        <BErWA>den Kopf b.</BErWA>
                        <BErWA>ich kann den Arm nicht mehr b.</BErWA>
                        <BErWA>der Wagen lässt sich nicht b.</BErWA>
                  </BFolgE>
                  <BFolgE>
                        <BNr>I.2.a</BNr>
                        <DefNr>1</DefNr>
                        <BGramAn>[V.1</BGramAn>
                        <BErkl BErklPhrase="innerlich berühren, in jmdm. ein
Gefühl erregen">
                              <notmatched Wort="innerlich"/>
                              <Verb Wort="berühren" Person="1. Pers. Plu./3.
Pers. Plu./Infinitiv" Tempus="Praesens" Praefix="not matched"/>
                              <Satzzeichen Wort="," Zeichen="Komma"/>
                              <Praeposition Wort="in" Modus="Temporal,Lokal,"/>
                              <Indefinitpronomen Wort="jemandem"
Kuerzel="jmdm." Kasus="Dat."/>
                              <Nominalphrase>
                                    <Artikel Wort="ein" Kasus="Nom./Nom./Akk."
```

```
Genus="Mask./Neut./Neut." Numerus="Sing./Sing./Sing." Art="unbestimmt"/>
                                    <Substantiv Wort="Gefühl" Genus="neutrum"
Kasus="Nom./Akk./Dat." Numerus="Sing./Sing./Sing."/>
                              </Nominalphrase>
                              <Verb Wort="erregen" Person="Infinitiv"
Tempus="Praesens" Praefix="not matched"/>
                        </BErkl>
                        <BErWA>seine Erzählung hat mich bewegt, es war ein ~der
Augenblick</BErWA>
                  </BFolgE>
                  <BFolgE>
                        <BNr>I..b</BNr>
                        <DefNr>2</DefNr>
                        <BGramAn>[V.1</BGramAn>
                        <BErkl BErklPhrase="innerlich verfolgen, beschäftigen,
in jmdm. Gedanken hervorrufen">
                              <notmatched Wort="innerlich"/>
                              <Verb Wort="verfolgen" Person="Infinitiv"
Tempus="Praesens" Praefix="not matched"/>
                              <Satzzeichen Wort="," Zeichen="Komma"/>
                              <Verb Wort="beschäftigen" Person="Infinitiv"
Tempus="Praesens" Praefix="not matched"/>
                              <Satzzeichen Wort="," Zeichen="Komma"/>
                              <Praeposition Wort="in" Modus="Temporal,Lokal,"/>
                              <Indefinitpronomen Wort="jemandem"
Kuerzel="jmdm." Kasus="Dat."/>
                              <Nominalphrase>
                                    <Substantiv Wort="Gedanke"
Genus="maskulin" Kasus="Nom./Akk./Akk./Dat./Dat./Gen."
Numerus="Plu./Sing./Plu./Sing./Plu./Plu."/>
                              </Nominalphrase>
                              <Verb Wort="hervorrufen" Person="1. Pers. Plu./3.
Pers. Plu./Infinitiv" Tempus="Praesens" Praefix="no"/>
                        </BErkl>
                        <BErWA>die Vorstellung eines Berufswechsels bewegt mich
schon lange</BErWA>
                  </BFolgE>
                  <BFolgE>
                        <BNr>I.3.</BNr>
                        <DefNr>3</DefNr>
                        <BGramAn>[V.1</BGramAn>
                        <BErkl BErklPhrase="es täglich reiten">
                              <Personalpronomen Wort="es" Kasus="Nom./Akk. "
Person="3 Pers./3 Pers." Numerus="Sing./Sing."/>
                              <Adverb Wort="täglich" Modus="Temporal,"/>
                              <Verb Wort="reiten" Person="1. Pers. Plu./3.
Pers. Plu./Infinitiv" Tempus="Praesens" Praefix="no"/>
                        </BErkl>
                        <BErWA/>
                  </BFolgE>
                  <BFolgE>
                        <BNr>II.1.</BNr>
                        <DefNr>4</DefNr>
                        <BGramAn>[V.1. hat bewegt</BGramAn>
                        <BErkl BErklPhrase="seine Lage, Stellung verändern">
                              <Nominalphrase>
                                    <Possessivpronomen Wort="seine" Person="3
Pers." Kasus="Nom./Nom./Akk./Akk." Numerus="Sing./Plu./Sing./Plu."/>
                                    <Substantiv Wort="Lage" Genus="feminin"
Kasus="Nom./Akk./Dat./Gen." Numerus="Sing./Sing./Sing./Sing."/>
                              </Nominalphrase>
                              <Satzzeichen Wort="," Zeichen="Komma"/>
                              <Nominalphrase>
                                    <Substantiv Wort="Stellung"
```

```
Genus="feminin" Kasus="Nom./Akk./Dat./Gen." Numerus="Sing./Sing./Sing./Sing."/>
                                </Nominalphrase>
                                <Verb Wort="verändern" Person="1. Pers. Plu./3.
Pers. Plu./Infinitiv" Tempus="Praesens" Praefix="no"/>
                          </BErkl>
                          <BErWA>ich kann mich vor Schmerzen kaum b.</BErWA>
                          <BErWA>beweg dich nicht!</BErWA>
                    </BFolgE>
                    <BFolgE>
                          <BNr>II.2.</BNr>
                          <DefNr>5</DefNr>
                          <BGramAn>[V.1. hat bewegt</BGramAn>
                          <BErkl BErklPhrase="seine Stellung fortgesetzt
verändern">
                                <Nominalphrase>
                                      <Possessivpronomen Wort="seine" Person="3
Pers." Kasus="Nom./Nom./Akk./Akk." Numerus="Sing./Plu./Sing./Plu."/>
                                      <Substantiv Wort="Stellung"
Genus="feminin" Kasus="Nom./Akk./Dat./Gen." Numerus="Sing./Sing./Sing./Sing."/>
                                </Nominalphrase>
                                <Verb Wort="setzen" Person="Infinitiv"
Tempus="Perfekt" Praefix="no"/>
                                <Verb Wort="verändern" Person="1. Pers. Plu./3.
Pers. Plu./Infinitiv" Tempus="Praesens" Praefix="no"/>
                          </BErkl>
                          <BErWA>beide Fahrzeuge b. sich in der gleichen
Richtung</BErWA>
                          <BErWA>ein langer Zug von Demonstranten bewegte sich
zum Rathaus</BErWA>
                    </BFolgE>
                    <BFolgE>
                          <BNr>II.3.</BNr>
                          <DefNr>6</DefNr>
                          <BGramAn>[V.1. hat bewegt</BGramAn>
                          <BErkl BErklPhrase="umhergehen, sich drehen, sich
bücken usw.">
                                <Nominalphrase>
                                      <Substantiv Wort="Umhergehen"
Genus="neutrum"/>
                                </Nominalphrase>
                                <Satzzeichen Wort="," Zeichen="Komma"/>
                                <Reflexivpronomen Wort="sich" Kasus="Akk."
Person="3 Pers./3 Pers." Numerus="Sing./Plu."/>
                                <Verb Wort="drehen" Person="1. Pers. Plu./3.
Pers. Plu./Infinitiv" Tempus="Praesens" Praefix="no"/>
                                <Satzzeichen Wort="," Zeichen="Komma"/>
                                <Reflexivpronomen Wort="sich" Kasus="Akk."
Person="3 Pers./3 Pers." Numerus="Sing./Plu."/>
                                <Verb Wort="bücken" Person="1. Pers. Plu./3.
Pers. Plu./Infinitiv" Tempus="Praesens" Praefix="no"/>
                                <Abkuerzung Wort="und so weiter" Kuerzel="usw."/>
                          </BErkl>
                          <BErWA>ich kann mich hier frei b.</BErWA>
                          <BErWA>sie bewegt sich sehr graziös</BErWA>
                          <BErWA>sich viel in frischer Luft b.</BErWA>
                          <BErWA>Bedeutung:</BErWA>
                          <BErWA>Umgang haben, verkehren</BErWA>
                          <BErWA>Bedeutung:</BErWA>
                          <BErWA>er bewegt sich in ganz anderen
Gesellschaftskreisen als ich</BErWA>
                    </BFolgE>
                    <BFolgE>
                          <BNr>III..</BNr>
                          <DefNr>7</DefNr>
```

```
                        <BGramAn>[V.11, hat bewogen</BGramAn>
                        <BErkl BErklPhrase="veranlassen, dazu bringen">
                              <Verb Wort="veranlassen" Person="1. Pers. Plu./3.
Pers. Plu./Infinitiv" Tempus="Praesens" Praefix="no"/>
                              <Satzzeichen Wort="," Zeichen="Komma"/>
                              <notmatched Wort="dazu"/>
                              <Verb Wort="bringen" Person="1. Pers. Plu./3.
Pers. Plu./Infinitiv" Tempus="Praesens" Praefix="no"/>
                        </BErkl>
                        <BErWA>er bewog mich, noch heute heimzufahren</BErWA>
                        <BErWA>ich konnte ihn nicht dazu b., sich zu
entschuldigen</BErWA>
                        <BErWA>was hat dich bewogen, diesen Hund zu kaufen?
</BErWA>
                  </BFolgE>
            </ArtEin>
      </Wartikel>

...

</BedWB>
```

C.4: Zusätzliche Verben für das Bedeutungswörterbuch

7.2006 – Prof. Dr. phil. G. Büchel:

antworten
beantworten
verantworten
sorgen
besorgen
versorgen
entsorgen
sagen
ansagen
absagen
besagen
entsagen
versagen
sickern
versickern
siegen
besiegen
entsiegen
fragen
befragen
hören
abhören
verhören
räuspern
husten
abhusten
anfragen
zuhören
anhören
prüfen
abprüfen
hüpfen
spielen
abspielen
anspielen
bespielen
überspielen
verspielen
überhören
überfragen
überantworten
setzen
absetzen
ansetzen
besetzen
entsetzen
übersetzen
versetzen
zusetzen
zusetzen
vorsetzen
stellen
abstellen
anstellen
bestellen
entstellen
überstellen
verstellen
vorstellen
zustellen
rücken
abrücken
entrücken
verrücken
senden
absenden
entsenden
versenden
überbrücken
steuern
ansteuern
besteuern
übersteuern
versteuern
würfeln
jagen
abjagen
bejagen
verjagen
fördern
befördern
meistern
reden
bereden
verabreden
überreden
trotten
abjagen
wenden
abwenden
anwenden
entwenden
verwenden
zusenden
wickeln
abwickeln
umwickeln
verwickeln
bohren
anbohren
aufbohren
aufhören
aufsagen
aufsetzen
aufstellen
aufspielen
aufrücken
aufwenden
kleben
ankleben
abkleben
verkleben
zukleben
überkleben
schrauben
anschrauben
abschrauben
verschrauben
drehen
andrehen
abdrehen
aufdrehen
verdrehen
überdrehen
bilden
abbilden
denken
erkennen
erwähnen
glauben
hoffen
meinen
verantworten
wissen

Anhang D: Treebolic Browser

D.1: DTD des Treebolic Browsers

Im Verzeichnis /treebolic/DOC/eg/animal (ausgehend vom Verzeichnis, in dem der Treebolic Browser[98] installiert wird):

```
<!-- DTD for treebolic v 1.7 20040410 -->

<!ELEMENT treebolic (tree,tools?)>
<!ATTLIST treebolic
     toolbar (true|false) #IMPLIED
     statusbar (true|false) #IMPLIED
     popupmenu (true|false) #IMPLIED
     tooltip (true|false) #IMPLIED
     focus-on-hover (true|false) #IMPLIED
     focus IDREF #IMPLIED
     xmoveto CDATA #IMPLIED
     ymoveto CDATA #IMPLIED
>

<!ELEMENT tree (img?,nodes,edges?)>
<!ATTLIST tree
     backcolor CDATA #IMPLIED
     forecolor CDATA #IMPLIED
     orientation CDATA #IMPLIED
     expansion CDATA #IMPLIED
     sweep CDATA #IMPLIED
     preserve-orientation (true|false) #IMPLIED
     fontface CDATA #IMPLIED
     fontsize CDATA #IMPLIED
     fontsizestep CDATA #IMPLIED
>

<!ELEMENT nodes (img?,node)>
<!ATTLIST nodes
     backcolor CDATA #IMPLIED
     forecolor CDATA #IMPLIED
>

<!ELEMENT node (label?,content?,img?,a?,mountpoint?,node*)>
<!ATTLIST node
     id ID #REQUIRED
     backcolor CDATA #IMPLIED
     forecolor CDATA #IMPLIED
>
```

98 Treebolic, http://sourceforge.net/projects/treebolic/ (2.2007)

```
<!ELEMENT label (#PCDATA)>

<!ELEMENT content (#PCDATA)>

<!ELEMENT a (#PCDATA)>
<!ATTLIST a
     href CDATA #REQUIRED
     target CDATA #IMPLIED
>

<!ELEMENT img (#PCDATA)>
<!ATTLIST img
     src CDATA #REQUIRED
>

<!ELEMENT mountpoint (a)>
<!ATTLIST mountpoint
     delaymount (true|false) #IMPLIED
     weight CDATA #IMPLIED
>

<!ELEMENT edges (edge*)>
<!ATTLIST edges
     treecolor CDATA #IMPLIED
     color CDATA #IMPLIED
     hyperbolic (true|false) #IMPLIED
>

<!ELEMENT edge (#PCDATA)>
<!ATTLIST edge
     from IDREF #REQUIRED
     to IDREF #REQUIRED
     color CDATA #IMPLIED
     hidden (true|false) #IMPLIED
>

<!ELEMENT tools (menu?)>
<!ATTLIST tools
>

<!ELEMENT menu (menuitem*)>
<!ATTLIST menu
>
<!ELEMENT menuitem (label, a?)>
<!ATTLIST menuitem
     action (goto|search|focus) #REQUIRED
     match-target CDATA #IMPLIED
     match-scope (label|content|link|id) #IMPLIED
     match-mode (equals|startswith|includes) #IMPLIED
>
```

D.2: XML-Datei für Treebolic Browser

Eine mit Hilfe der Anwendung SentenceView generierte XML-Datei, die der DTD des Treebolic Browsers unterliegt:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE treebolic SYSTEM "Treebolic.dtd">
<!--treebolic ï¿½ Sun May 30 07:53:25 CEST 2004-->
<treebolic toolbar="true" statusbar="true" popupmenu="true" tooltip="true"
focus-on-hover="false">
 <tree orientation="radial" expansion="0.9" sweep="1.2" backcolor="ffffff"
forecolor="000000" fontface="Tahoma Gras" fontsize="20" fontsizestep="2"
preserve-orientation="true">
  <img src="backgrnd.gif"/>
   <nodes backcolor="9999ff" forecolor="ffffff">
    <img src="item.gif"/>

     <node id="8872">
      <img src="Root.gif"/>
      <node id="7986">
       <img src="Satz.gif"/>
       <node id="7987">
        <img src="Verb.gif"/>
        <node id="7989">
         <img src="Bedeutung.gif"/>
         <node id="7990">
          <img src="Praepositionalphrase.gif"/>
          <node id="7991">
           <img src="Praeposition.gif"/>
           <label>Wort: "mit", Modus: "Modal,Instrumental,", </label>
          </node>
          <node id="7992">
           <img src="Nominalphrase.gif"/>
           <node id="7993">
            <img src="Artikel.gif"/>
            <label>Wort: "der", Kasus: "Nom./Gen./Dat./Gen.", Art: "bestimmt",
Numerus: "Sing./Sing./Sing./Plu.", Genus: "Mask./Fem./Fem./*/", </label>
           </node>
           <node id="7994">
            <img src="Substantiv.gif"/>
            <label>Wort: "Hand", Kasus: "Nom./Akk./Dat./Gen.", Numerus:
"Sing./Sing./Sing./Sing.", Genus: "feminin", </label>
           </node>
           <label></label>
          </node>
          <label></label>
         </node>
         <node id="7995">
          <img src="Satzzeichen.gif"/>
          <label>Wort: ",", Zeichen: "Komma", </label>
         </node>
         <node id="7996">
          <img src="Nominalphrase.gif"/>
          <node id="7997">
           <img src="Artikel.gif"/>
           <label>Wort: "den", Kasus: "Akk./Dat.", Art: "bestimmt", Numerus:
"Sing./Plu.", Genus: "Mask./*", </label>
          </node>
          <node id="7998">
           <img src="Substantiv.gif"/>
```

```
           <label>Wort: "Finger", Kasus: "Dat.", Numerus: "Plu.", Genus:
"maskulin", </label>
          </node>
          <label></label>
         </node>
         <node id="7999">
          <img src="Partizip.gif"/>
          <label>Wort: "fühlen", Steigerungsform: "Positiv", </label>
         </node>
         <node id="8000">
          <img src="Verb.gif"/>
          <node id="8041">
           <img src="Bedeutung.gif"/>
           <node id="8042">
            <img src="notmatched.gif"/>
            <label>Wort: "innerlich", </label>
           </node>
           <node id="8043">
            <img src="Verb.gif"/>
            <label>Wort: "bewegen", Person: "1. Pers. Plu./3. Pers.
Plu./Infinitiv", Praefix: "no", Tempus: "Praesens", </label>
           </node>
           <label>BNr: "2.", BErklPhrase: "innerlich bewegen", </label>
          </node>
          <label>Wort: "berühren", Person: "1. Pers. Plu./3. Pers.
Plu./Infinitiv", Praefix: "no", Tempus: "Praesens", </label>
         </node>
         <node id="8001">
          <img src="Satzzeichen.gif"/>
          <label>Wort: ",", Zeichen: "Komma", </label>
         </node>
         <node id="8002">
          <img src="Verb.gif"/>
          <node id="8088">
           <img src="Bedeutung.gif"/>
           <node id="8089">
            <img src="notmatched.gif"/>
            <label>Wort: "zufällig", </label>
           </node>
           <node id="8090">
            <img src="Konjunktion.gif"/>
            <label>Wort: "oder", Modus: "disjunktiv", </label>
           </node>
           <node id="8091">
            <img src="Praepositionalphrase.gif"/>
            <node id="8092">
             <img src="Praeposition.gif"/>
             <label>Wort: "durch", Modus: "Instrumental,Lokal,Kausal,",
</label>
            </node>
            <node id="8093">
             <img src="Nominalphrase.gif"/>
             <node id="8094">
              <img src="Substantiv.gif"/>
              <label>Wort: "Suche", Kasus: "Nom./Akk./Dat./Gen.", Numerus:
"Plu./Plu./Plu./Plu.", Genus: "feminin", </label>
             </node>
             <label></label>
            </node>
            <label></label>
           </node>
           <node id="8095">
            <img src="Adjektiv.gif"/>
```

```
            <label>Wort: "gewahr", Steigerungsform: "Positiv", </label>
           </node>
           <node id="8096">
            <img src="Hilfsverb.gif"/>
            <label>Wort: "werden", Infinitiv: "werden", </label>
           </node>
           <node id="8097">
            <img src="Satzzeichen.gif"/>
            <label>Wort: ",", Zeichen: "Komma", </label>
           </node>
           <node id="8099">
            <img src="Satzzeichen.gif"/>
            <label>Wort: ",", Zeichen: "Komma", </label>
           </node>
           <node id="8100">
            <img src="Verb.gif"/>
            <label>Wort: "sehen", Person: "1. Pers. Plu./3. Pers.
Plu./Infinitiv", Praefix: "no", Tempus: "Praesens", </label>
           </node>
           <label>BNr: "I.1.a", BErklPhrase: "zufällig oder durch Suchen gewahr
werden, entdecken, sehen", </label>
          </node>
          <label>Wort: "finden", Person: "1. Pers. Plu./3. Pers.
Plu./Infinitiv", Praefix: "no", Tempus: "Praesens", </label>
         </node>
         <label>BNr: "I.1.", BErklPhrase: "mit der Hand, den Fingern fühlend
berühren, finden", </label>
        </node>
        <node id="8132">
         <img src="Bedeutung.gif"/>
         <node id="8133">
          <img src="Praepositionalphrase.gif"/>
          <node id="8134">
           <img src="Praeposition.gif"/>
           <label>Wort: "mit", Modus: "Modal,Instrumental,", </label>
          </node>
          <node id="8135">
           <img src="Nominalphrase.gif"/>
           <node id="8136">
            <img src="Artikel.gif"/>
            <label>Wort: "den", Kasus: "Akk./Dat.", Art: "bestimmt", Numerus:
"Sing./Plu.", Genus: "Mask./*", </label>
           </node>
           <node id="8137">
            <img src="Substantiv.gif"/>
            <label>Wort: "Hand", Kasus: "Dat.", Numerus: "Plu.", Genus:
"feminin", </label>
           </node>
           <label></label>
          </node>
          <label></label>
         </node>
         <node id="8138">
          <img src="Partizip.gif"/>
          <label>Wort: "berühren", Steigerungsform: "Positiv", </label>
         </node>
         <node id="8139">
          <img src="Konjunktion.gif"/>
          <label>Wort: "und", Modus: "kopulativ", </label>
         </node>
         <node id="8140">
          <img src="Praeposition.gif"/>
          <label>Wort: "ohne", Modus: "Kausal,Modal,", </label>
```

```
          </node>
          <node id="8141">
           <img src="Praeposition.gif"/>
           <label>Wort: "zu", Modus: "Temporal,Lokal,", </label>
          </node>
          <node id="8142">
           <img src="Verb.gif"/>
           <label>Wort: "sehen", Person: "1. Pers. Plu./3. Pers.
Plu./Infinitiv", Praefix: "no", Tempus: "Praesens", </label>
          </node>
          <node id="8143">
           <img src="Reflexivpronomen.gif"/>
           <label>Wort: "sich", Person: "3 Pers./3 Pers.", Kasus: "Akk.",
Numerus: "Sing./Plu.", </label>
          </node>
          <node id="8144">
           <img src="notmatched.gif"/>
           <label>Wort: "vorwärts", </label>
          </node>
          <node id="8145">
           <img src="Verb.gif"/>
           <label>Wort: "bewegen", Person: "1. Pers. Plu./3. Pers.
Plu./Infinitiv", Praefix: "no", Tempus: "Praesens", </label>
          </node>
          <node id="8146">
           <img src="Nominalphrase.gif"/>
           <node id="8147">
            <img src="Substantiv.gif"/>
            <label>Wort: "Gegenstand", Kasus: "Gen./Nom./Dat./Gen.", Numerus:
"Plu./Plu./Plu./Plu.", Genus: "maskulin", </label>
           </node>
           <label></label>
          </node>
          <node id="8148">
           <img src="Partizip.gif"/>
           <label>Wort: "berühren", Steigerungsform: "Positiv", </label>
          </node>
          <node id="8149">
           <img src="Konjunktion.gif"/>
           <label>Wort: "und", Modus: "kopulativ", </label>
          </node>
          <node id="8150">
           <img src="Praeposition.gif"/>
           <label>Wort: "ohne", Modus: "Kausal,Modal,", </label>
          </node>
          <node id="8151">
           <img src="Praeposition.gif"/>
           <label>Wort: "zu", Modus: "Temporal,Lokal,", </label>
          </node>
          <node id="8152">
           <img src="Verb.gif"/>
           <label>Wort: "sehen", Person: "1. Pers. Plu./3. Pers.
Plu./Infinitiv", Praefix: "no", Tempus: "Praesens", </label>
          </node>
          <node id="8153">
           <img src="Reflexivpronomen.gif"/>
           <label>Wort: "sich", Person: "3 Pers./3 Pers.", Kasus: "Akk.",
Numerus: "Sing./Plu.", </label>
          </node>
          <node id="8154">
           <img src="notmatched.gif"/>
           <label>Wort: "vorwärts", </label>
          </node>
```

```
         <node id="8155">
          <img src="Verb.gif"/>
          <label>Wort: "bewegen", Person: "1. Pers. Plu./3. Pers.
Plu./Infinitiv", Praefix: "no", Tempus: "Praesens", </label>
         </node>
         <label>BNr: "II..", BErklPhrase: "mit den Händen Gegenstände berührend
und ohne zu sehen sich vorwärts bewegen", </label>
        </node>
        <node id="8497">
         <img src="Bedeutung.gif"/>
         <node id="8498">
          <img src="Praepositionalphrase.gif"/>
          <node id="8499">
           <img src="Praeposition.gif"/>
           <label>Wort: "mit", Modus: "Modal,Instrumental,", </label>
          </node>
          <node id="8500">
           <img src="Nominalphrase.gif"/>
           <node id="8501">
            <img src="Artikel.gif"/>
            <label>Wort: "der", Kasus: "Nom./Gen./Dat./Gen.", Art: "bestimmt",
Numerus: "Sing./Sing./Sing./Plu.", Genus: "Mask./Fem./Fem./*/", </label>
           </node>
           <node id="8502">
            <img src="Substantiv.gif"/>
            <label>Wort: "Hand", Kasus: "Nom./Akk./Dat./Gen.", Numerus:
"Sing./Sing./Sing./Sing.", Genus: "feminin", </label>
           </node>
           <label></label>
          </node>
          <label></label>
         </node>
         <node id="8503">
          <img src="Satzzeichen.gif"/>
          <label>Wort: ",", Zeichen: "Komma", </label>
         </node>
         <node id="8504">
          <img src="Praepositionalphrase.gif"/>
          <node id="8505">
           <img src="Praeposition.gif"/>
           <label>Wort: "mit", Modus: "Modal,Instrumental,", </label>
          </node>
          <node id="8506">
           <img src="Nominalphrase.gif"/>
           <node id="8507">
            <img src="Artikel.gif"/>
            <label>Wort: "den", Kasus: "Akk./Dat.", Art: "bestimmt", Numerus:
"Sing./Plu.", Genus: "Mask./*", </label>
           </node>
           <node id="8508">
            <img src="Substantiv.gif"/>
            <label>Wort: "Hand", Kasus: "Dat.", Numerus: "Plu.", Genus:
"feminin", </label>
           </node>
           <label></label>
          </node>
          <label></label>
         </node>
         <node id="8509">
          <img src="Satzzeichen.gif"/>
          <label>Wort: ",", Zeichen: "Komma", </label>
         </node>
         <node id="8510">
```

```
          <img src="Praepositionalphrase.gif"/>
          <node id="8511">
           <img src="Praeposition.gif"/>
           <label>Wort: "mit", Modus: "Modal,Instrumental,", </label>
          </node>
          <node id="8512">
           <img src="Nominalphrase.gif"/>
           <node id="8513">
            <img src="Artikel.gif"/>
            <label>Wort: "den", Kasus: "Akk./Dat.", Art: "bestimmt", Numerus:
"Sing./Plu.", Genus: "Mask./*", </label>
           </node>
           <node id="8514">
            <img src="Substantiv.gif"/>
            <label>Wort: "Finger", Kasus: "Dat.", Numerus: "Plu.", Genus:
"maskulin", </label>
           </node>
           <label></label>
          </node>
          <label></label>
         </node>
         <node id="8515">
          <img src="Satzzeichen.gif"/>
          <label>Wort: ",", Zeichen: "Komma", </label>
         </node>
         <node id="8516">
          <img src="Praepositionalphrase.gif"/>
          <node id="8517">
           <img src="Praeposition.gif"/>
           <label>Wort: "mit", Modus: "Modal,Instrumental,", </label>
          </node>
          <node id="8518">
           <img src="Nominalphrase.gif"/>
           <node id="8519">
            <img src="Artikel.gif"/>
            <label>Wort: "einem", Kasus: "Dat./Dat.", Art: "unbestimmt",
Numerus: "Sing./Sing.", Genus: "Fem./Neut.", </label>
           </node>
           <node id="8520">
            <img src="Substantiv.gif"/>
            <label>Wort: "Gegenstand", Kasus: "Nom./Akk./Dat.", Numerus:
"Sing./Sing./Sing.", Genus: "maskulin", </label>
           </node>
           <label></label>
          </node>
          <label></label>
         </node>
         <node id="8521">
          <img src="Satzzeichen.gif"/>
          <label>Wort: ")", Zeichen: "rundeKlammerzu", </label>
         </node>
         <node id="8522">
          <img src="Verb.gif"/>
          <label>Wort: "suchen", Person: "1. Pers. Plu./3. Pers.
Plu./Infinitiv", Praefix: "no", Tempus: "Praesens", </label>
         </node>
         <label>BNr: "III.1.", BErklPhrase: "mit der Hand, mit den Händen, mit
den Fingern, mit einem Gegenstand) suchen", </label>
        </node>
        <node id="8597">
         <img src="Bedeutung.gif"/>
         <node id="8598">
          <img src="Praeposition.gif"/>
```

```
          <label>Wort: "zu", Modus: "Temporal,Lokal,", </label>
         </node>
         <node id="8600">
          <img src="Verb.gif"/>
          <label>Wort: "suchen", Person: "1. Pers. Plu./3. Pers.
Plu./Infinitiv", Praefix: "no", Tempus: "Praesens", </label>
         </node>
         <label>BNr: "III.2.", BErklPhrase: "zu erfahren suchen", </label>
        </node>
        <label>Wort: "tasten", Person: "1. Pers. Plu./3. Pers. Plu./Infinitiv",
Tempus: "Praesens", Praefix: "no", </label>
       </node>
       <label></label>
      </node>
      <node id="8670">
       <img src="Satz.gif"/>
       <node id="8671">
        <img src="Verb.gif"/>
        <node id="8822">
         <img src="Bedeutung.gif"/>
         <node id="8823">
          <img src="Satzzeichen.gif"/>
          <label>Wort: "(", Zeichen: "rundeKlammerauf", </label>
         </node>
         <node id="8824">
          <img src="Praepositionalphrase.gif"/>
          <node id="8825">
           <img src="Praeposition.gif"/>
           <label>Wort: "mit", Modus: "Modal,Instrumental,", </label>
          </node>
          <node id="8826">
           <img src="Nominalphrase.gif"/>
           <node id="8827">
            <img src="Substantiv.gif"/>
            <label>Wort: "Hilfe", Kasus: "Nom./Akk./Dat./Gen.", Numerus:
"Sing./Sing./Sing./Sing.", Genus: "feminin", </label>
           </node>
           <label></label>
          </node>
          <label></label>
         </node>
         <node id="8828">
          <img src="Nominalphrase.gif"/>
          <node id="8829">
           <img src="Artikel.gif"/>
           <label>Wort: "des", Kasus: "Gen./Gen.", Art: "bestimmt", Numerus:
"Sing./Sing.", Genus: "Mask./Neut.", </label>
          </node>
          <node id="8830">
           <img src="Substantiv.gif"/>
           <label>Wort: "Geruchssinn", Kasus: "Gen.", Numerus: "Sing.", Genus:
"maskulin", </label>
          </node>
          <label></label>
         </node>
         <node id="8831">
          <img src="Satzzeichen.gif"/>
          <label>Wort: ")", Zeichen: "rundeKlammerzu", </label>
         </node>
         <node id="8832">
          <img src="Nominalphrase.gif"/>
          <node id="8833">
           <img src="Artikel.gif"/>
```

```
           <label>Wort: "eine", Kasus: "Nom./Akk.", Art: "unbestimmt", Numerus:
"Sing./Sing.", Genus: "Fem./Fem.", </label>
          </node>
          <node id="8834">
           <img src="Substantiv.gif"/>
           <label>Wort: "Fährte", Kasus: "Nom./Akk./Dat./Gen.", Numerus:
"Sing./Sing./Sing./Sing.", Genus: "feminin", </label>
          </node>
          <label></label>
         </node>
         <node id="8836">
          <img src="Konjunktion.gif"/>
          <label>Wort: "oder", Modus: "disjunktiv", </label>
         </node>
         <node id="8837">
          <img src="Nominalphrase.gif"/>
          <node id="8838">
           <img src="Artikel.gif"/>
           <label>Wort: "eine", Kasus: "Nom./Akk.", Art: "unbestimmt", Numerus:
"Sing./Sing.", Genus: "Fem./Fem.", </label>
          </node>
          <node id="8839">
           <img src="Substantiv.gif"/>
           <label>Wort: "Fährte", Kasus: "Nom./Akk./Dat./Gen.", Numerus:
"Sing./Sing./Sing./Sing.", Genus: "feminin", </label>
          </node>
          <label></label>
         </node>
         <node id="8840">
          <img src="Verb.gif"/>
          <label>Wort: "suchen", Person: "1. Pers. Plu./3. Pers.
Plu./Infinitiv", Praefix: "not matched", Tempus: "Praesens", </label>
         </node>
         <label>BNr: "II..", BErklPhrase: "(mit Hilfe des Geruchssinnes) eine
Fährte verfolgen oder eine Fährte suchen", </label>
        </node>
        <label>Wort: "spüren", Person: "1. Pers. Plu./3. Pers. Plu./Infinitiv",
Tempus: "Praesens", Praefix: "not matched", </label>
       </node>
       <label></label>
      </node>
     </node>
    </nodes>

    <edges treecolor="993300" color="ff0033" hyperbolic="true">        <edge
from="8152" to="8142" color="0033ff"/>
      <edge from="8155" to="8145" color="0033ff"/>
      <edge from="8600" to="8522" color="0033ff"/>
      <edge from="8840" to="8522" color="0033ff"/>
    </edges>
   </tree>
</treebolic>
```

Anhang E : Anmerkungen zum Programmcode

Die Anhänge E bis I sind nicht in der veröffentlichten Ausgabe dieser Masterarbeit enthalten (siehe „Änderungen und Einschränkungen der KOPS-Veröffentlichung aus Gründen des Copyrights“).

Diese Anhänge tragen in der ursprünglichen Fassung folgende Kapitelnamen:

Anhang F : Schnittstellen von IPEE

Anhang G : Java Source-Code

Anhang H : Threadlist-XML-Dateien

Anhang I : Patternlist-XML-Dateien