Fachbereich Informatik
Institut für Computervisualistik

DIPLOMARBEIT

Evaluation von Syntaxanalysemethoden angewendet auf Datenmaterial aus transliterierten Schulaufsätzen

Eingereicht von: Fränkel, Caroline
Im blauen Garn 83
50389 Wesseling
geboren am: 28.05.1980 in Limburg

Studiengang: Computervisualistik

Betreuer: Prof. Dr. rer. nat. Karin Harbusch
Diplom-Informatiker Ulrich Koch

Eidesstattliche Erklärung

Ich erkläre an Eides statt, gegenüber der Universität Koblenz, dass ich die vorliegende Diplomarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel gefertigt habe. Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keinem anderen Prüfer als Prüfungsleistung eingereicht.

Datum: _____ Unterschrift: _____

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Diplomarbeit beigetragen haben.
Besonders möchte ich mich bei meinen beiden Korrektoren Karin Harbusch und Uli Koch bedanken, die mich durch hilfreiche Anregungen und ihre Geduld immer wieder unterstützt haben.

Ich danke allen Freunden und Bekannten, die mir in jeder Phase der Diplomarbeit ihre moralische Unterstützung haben zukommen lassen.

Ganz besonderer Dank geht hierbei an Stefan Schneider, der sich besonders in der schwierigen Schlussphase dieser Arbeit als gute und geduldige Begleitung und Hilfe erwies und darüber hinaus als Lektor zur Verfügung stand.

Über allem stehen natürlich meine Eltern, ohne deren moralische und finanzielle Unterstützung, dieses Studium nie möglich gewesen wäre.

„...there is no such thing as human nature independent of culture“ (Geertz, 1973)

Inhaltsverzeichnis

Kurzzeichenverzeichnis

Kurzzeichen	Bezeichnung
NLP	Natural Language Processing
SMES	Saarbrücker Message Extraction System
BC	Base-Clause
MC	Main Clause
VF	Verb Fragment
HPSG	Head-driven Phrase Structure Grammar
NP	Nominalphrase
VP	Verbalphrase
PP	Präpositionalphrase
DFKI	Deutsches Forschungsinstitut für Künstliche Intelligenz
FST	Finite State Transductor
VVIMP	Verb im Imperativ, voll
PPER	Irreflexibles Personalpronomen
APB	Average Parse Base
ARTDEF	Bestimmter Artikel
NN	Normales Nomen
FCP	Fragment Combination Pattern
FST	Finite State Transducer

1 Einleitung

1.1 Motivation

Computergestütztes Lernen, auch E-Learning oder tutorielle Systeme genannt, ist ein Spezialfall natürlichen Lernens [DIT03] und [EUL04] und hält immer mehr Einzug in deutschen Schulen. Bereits in der Grundschule erlernen Kinder in Deutschland den Umgang mit Computern [MIT98].

Im Falle von computergestütztem Lernen interagieren die Lernenden mit einer Maschine. Die Qualität eines Lernprogramms hängt deshalb immer mit der Qualität der Interaktion mit dem Computer ab. Außer Frage steht, dass eine Maschine, in gewissen Umgebungen, einen menschlichen Lehrer/Lehrerin nicht wird ersetzten können. Jedoch ist es durchaus möglich, sich die Vorteile eines Computers derart zu Nutze zu machen, dass man dem Lehrer/der Lehrerin eine Hilfestellung und Arbeits- beziehungsweise Zeitersparnis bieten kann. Eine Bewertung der Fakten beim Lernen ist ein äußerst wichtiger Teil innerhalb eines Lernprozesses. Speziell innerhalb des Prozesses, Kindern in der Grundschule das Schreiben beizubringen, spielt die Form und Art der Bewertung eine wichtige Rolle. Kindern in diesem Alter und in dieser Stufe eines Lernprozesses, indem es darum geht, eine noch nicht vorhandene Fähigkeit Stück für Stück so zu erlernen, so dass sie anschließend in der Lage sind, fehlerfrei und sicher in der deutschen Sprache Texte zu schreiben, kann man nicht die Art und Form von Bewertung entgegen bringen, wie man es vielleicht mit GymnasiastInnen oder gar StudentInnen machen würde. Man muss die Arbeit und die Lernfortschritte der Kinder so bewerten, dass sie verstehen, wo eventuelle Fehler in ihrem Gedankenprozess liegen, dass sie verstehen, wie die verschiedenen Grammatik- oder Satzzeichenregeln, zum Beispiel, angewandt werden, und man sollte natürlich darauf achten, dass die Bewertung in einer Art und Weise stattfindet, die die Motivation und das Selbstbewusstsein der Kinder nicht negativ beeinflusst. Darüber hinaus soll durch die Hilfestellung des Computers es jeder Lehrkraft die Möglichkeit zur Auseinandersetzung mit der eigenen Diagnosefähigkeit gegeben werden [VER01]. Die Aufgabenstellungen, mit denen die Kinder in der Grundschule konfrontiert werden, unterliegen konkreten Lehrplänen.

Den Kindern werden Aufgaben gestellt, die sich nach einem gewissen Bildungsstandard richten und nach [VER01] folgende inhaltsbezogene Kompetenzen abdecken:

- Lesen – mit Texten und Medien umgehen,
- Sprache und Sprachgebrauch untersuchen,
- Schreiben (Texte verfassen und richtig schreiben).

Durch diese angestrebten Kompetenzen ergibt sich der Hauptgedanke dieser Diplomarbeit, die grammatikalische Korrektheit der Sätze in den vorliegenden Kindergeschichten zu bestimmen, die häufigsten Fehler aufzuzeigen, und herauszufinden, welche linguistischen Ansätze, Methoden und Verfahren sich im Umgang und der Verarbeitung dieser Texte als nützlich erweisen, um im größeren Kontext, der Entwicklung einer Hilfestellung für Lehrkräfte, brauchbare Aussagen über die Qualität der vorliegenden Schreibproben machen zu können. Die Frage die sich stellt ist wie weit man mit einer speziellen Art linguistischer Werkzeuge, angewandt auf die Kindergeschichten, kommt. Anschließend soll heraus gefunden werden, welche einfachen Methoden zur Verbesserung des Ergebnisses beitragen können. Nicht zuletzt soll dadurch idealerweise unter anderem eine Zeit- und Arbeitsersparnis erreicht werden, die es den Lehrkräften möglich macht, diese gewonnene Zeit anderweitig einzusetzen und damit einem übergeordnetem Fernziel, wie der Verbesserung der Schul- und Unterrichtsentwicklung näher zu kommen.

Um ein System zu erhalten, das dem angestrebten Ziel, einer Unterstützung der Lehrerinnen und Lehrer, so nahe wie möglich kommt, ist es naheliegend, ein bereits vorhandenes oder in sich in der Entwicklung befindendes System, der gegebenen beziehungsweise geplanten Lernumgebung und vor allen Dingen dem geplanten Lernziel bezüglich, bewerten zu können. Die Technik der flachen und partiellen Analyse hat einen besonderen Status gewonnen. Partielle Analyse ist robust und fehlertolerant. Für diese Art der Analyse ist kein semantisches Wissen erforderlich. Deshalb ist die allgemeine Vorhersage, dass sie sich für die Anwendung im Rahmen dieser Diplomarbeit gut eignet. Im Laufe der Entwicklung der Verarbeitung natürlicher Sprache ist eine Vielzahl von Parsern entstanden und genutzt worden. Mehr Informationen dazu werden in Kapitel 3.3 Parser erläutert. Parsing ist eine essentielle Verarbeitungsstufe in Natural Language Processing-Systemen. Von einem guten Parser wird erwartet, dass er die genaue syntaktische Struktur, das bedeutet die Form und Struktur, eines Satzes liefert. Die Leistungsfähigkeit und Qualität eines NLP-Systems hängt also in hohem Maß von der Leistungsfähigkeit und Qualität des gegebenen Parsers ab.

Dadurch werden systematische Methoden für eine breite Evaluation, der Leistungsfähigkeit dieser Programme immer notwendiger. Vergleichbare und reproduzierbare Evaluationen und Experiment sind wichtige Quellen für die Forschung.

Der Wunsch nach einem wenigstens teilweise automatisierten Prozess ist groß, da man es in der Regel auf diesem Gebiet der Computerlinguistik mit einem großen Bestand an Daten und damit auch an linguistischen Problemstellungen zu tun hat.

Die Evaluation eines Systems hängt immer von einer Vielzahl von Faktoren ab und kann nicht immer nach dem selben Prinzip erfolgen. Man ist sich einig, dass die Notwendigkeit besteht, geeignete Methoden und Verfahren zu finden, um die verschiedenen Systeme evaluieren zu können, gleichzeitig ist jedoch klar, dass es keine spezifische Methode gibt, die sich auf jedes System anwenden lässt. Die Entwicklung von Methoden für die automatische Evaluation von Parsern ist deshalb eine interessante Aufgabe, da sich immer wieder neue Aufgaben stellen, die es der gegebenen Situation entsprechend bestmöglich zu lösen gilt.

Diese Arbeit beschäftigt sich speziell mit der Evaluation eines bereits existierenden Chunk-Parsers im Kontext mit einer sehr speziellen Art von Texten. Es handelt sich um Schreibproben von Grundschulkindern. Die Frage ist, inwieweit ein man den Chunk-Parser als linguistisches Mittel nutzen kann, um die gegebenen Texte sinnvoll zu verarbeiten. Ist es eventuell sogar möglich, die gewonnen Informationen dazu zu nutzen, ein Langzeitziel wie zum Beispiel die Entwicklung eines computergestützten Lernsystems zur Unterstützung und Entlastung von Lehrern zu realisieren.

Die vorliegende Arbeit beschäftigt sich im Kern mit linguistischer Syntax. Es handelt sich also um eine wissenschaftliche Untersuchung der Form und Struktur natürlicher Sprache. Die Syntax ist der Teil der Grammatik, der sich mit dem Bau und der Gliederung des Satzes beschäftigt [MEY07].

1.2 Problemstellung

Im Rahmen dieser Diplomarbeit, wurden Texte untersucht, die von Grundschulkindern unter bestimmten Bedingungen und Voraussetzungen geschrieben wurden.
Die Texte entstanden im Rahmen des Projektes VERA (Vergleichsarbeiten in der Grundschule), das von Prof. Dr. Andreas Helmke und Juniorprof. Dr. Ingmar Hosenfeld durchgeführt wird [VER06]. Es wurden circa 1000 handgeschriebene Geschichten transliteriert und teilweise korrigiert. Nähere Informationen zur Entstehung und Bearbeitung der Texte sind in Kapitel 4 zu finden.

1.3 Aufbau der Arbeit

Für diese Diplomarbeit wurden die Texte mit dem Saarbrücker Message Extraction System (SMES), der am Deutschen Forschungsinstitut für Künstliche Intelligenz (DFKI) [NEU97] entwickelt wurde, verarbeitet. Zusätzlich wurden die Texte einer Analyse von Hand unterzogen, um eine Aussage über die Qualität von SMES machen zu können. Die vorliegende Diplomarbeit beschreibt die Konzeption des Parsingansatzes und eine durchgeführte Evaluation. Außerdem werden Vorschläge für einfache und sinnvolle Verbesserungen und Änderungen gemacht, die für den gegebenen Korpus sinnvoll erscheinen.

Ziel dieser Arbeit ist es, zu zeigen, welche Arbeits- und Verarbeitungsschritte notwendig und sinnvoll sind, um anschließend eine Aussage darüber treffen zu können, welche computerlinguistischen Methoden sich eignen, um die Entscheidung treffen zu können, welche Module man entwickeln kann, um den Lehrern und Schülern eine adäquate Lernhilfe zur Verfügung stellen zu können. Die Herausforderung bestand darin, zunächst ein linguistisches Mittel zu finden, das in Bezug auf die vorliegende Textart als am besten geeignet erschien und diese Wahl zu begründen. Anschließend galt es die Arbeitsweise und die Resultate der getroffenen Wahl genau zu untersuchen und heraus zu finden, welche einfachen Modifikationen man in das bereits bestehende System einbetten kann, um das Ergebnis weiter zu verbessern.

Die Arbeit gliedert sich in die folgenden Kapitel:

- In **Kapitel 2** wird zunächst der größere Kontext, in den diese Arbeit eingebettet ist, umrissen. Auf der Grundlage der unterschiedlichen Ansätze zur Verarbeitung natürlicher Sprache, die hier kurz vorgestellt werden, wird ein geeigneter Ansatz für die spezielle vorliegende Textsorte der Kinderaufsätze ausgewählt.
- In **Kapitel 3** erläutert die einzelnen Komponenten und die Systemarchitektur von SMES. Es wird geschildert, welche sprachlichen Phänomene sich damit verarbeiten lassen, mit welchen Problemen zu rechnen ist und wie die Ausgabe gestaltet ist.
- **Kapitel 4** wird zunächst näher auf die Entstehung des Evaluationskorpus eingegangen. Die Leistungsfähigkeit des Parsers in Bezug auf die Art der vorliegenden Texte wird durch eine Evaluation demonstriert. Es geht dabei sowohl um die Leistung ohne, als auch um die Leistung nach einfachen vorgenommenen Änderungen.
- In **Kapitel 5** fasst die gesammelten Erkenntnisse zusammen und liefert einen Ausblick auf eventuelle zukünftige Entwicklungen.

2 Praktischer und theoretischer Hintergrund zum syntaktischen Parsen

Eine Aufgabe der Computerlinguistik ist das Entwickeln von Computerprogrammen, mit deren Hilfe natürliche Sprache verarbeitet werden kann. Im folgenden soll näher auf die Verarbeitung natürlicher Sprache, das Natural Language Processing (NLP) und die Bedeutung der Methoden und Verfahren zur automatischen Verarbeitung natürlicher Sprache eingegangen werden. Später wird auf die Analyse, das heißt das Parsing eingegangen. Der Generierungsprozess wird in dieser Arbeit nicht behandelt.

Dazu muss man zunächst zwischen den drei überlappenden Bereichen Syntax, Semantik und Pragmatik unterscheiden. Diese Bereiche werden durch ihre Beziehungen zwischen Zeichen, der Zeichenbedeutung und den Benutzern der Zeichen in einer bestimmten Situation definiert.

Die Syntax befasst sich mit der formalen und nicht der inhaltlichen Struktur der Ausdrücke einer Sprache [SCH02]. Die Syntax bezeichnet die „Beziehungen, die sich aus der Anordnung der Zeichen ergeben“ [GOO02]. Goos führt dazu folgendes Beispiel an:

„das Haus“: Folge von sieben Buchstaben und einem Zwischenraum

Das ist eher die Sicht der Formalen Sprachen auf eine Zeichenkette. Die linguistische Syntax, mit der sich diese Arbeit im folgenden beschäftigt, spricht hier von einer Nominalphrase (NP).

Die Semantik hingegen befasst sich mit der inhaltlichen (Bedeutungs-) Struktur der Ausdrücke einer Sprache [SCH02]. Die Semantik bezeichnet nach [GOO02] die „Beziehungen, die sich aus der Interpretationsvorschrift ergeben“. Für das angeführte Beispiel bedeutet das:

„das Haus“: Zwischenraum trennt Wörter, also zwei Wörter

Die Pragmatik ist eine linguistische Disziplin, die sprachliches Handeln und die Verwendung von Sprache erforscht [WIK02]. [GOO02] bezeichnet das als die „Beziehung zwischen Daten und Gegenständen bzw. Sachverhalten außerhalb der Datenmenge“. Bezogen auf das bereits angeführte Beispiel bedeutet dies nach [GOO02]:

„das Haus“: Begriff für einen Gegenstand der realen Welt

Abbildung 2.1 stellt die Bedeutung von Syntax und Semantik in natürlichen Sprachen dar.

Abb. 2.1: Syntax und Semantik in natürlichen Sprachen

Die Syntax beschäftigt sich demnach mit der Form und Struktur natürlicher Sprache. Die linguistische Semantik hingegen, befasst sich primär mit dem Sinn. Im Kontext der vorliegenden Diplomarbeit beschäftige ich mich deshalb ausschließlich mit der Syntax natürlicher Sprache.
Im einzelnen werden verschiedene Verfahren spezifischer erläutert, die im Kontext dieser Arbeit eine wichtige Rolle spielen.
Diese Verfahren sind das *Part of Speech Tagging (POS), Chunk Parsing* und *Shallow Parsing.*

2.1 Natürlichsprachliche Verarbeitung (Natural Language Processing; NLP)

Natural Language Processing ist ein spezielles Thema im Bereich der Künstlichen Intelligenz (KI) und der Linguistik [JUR00] und [CAR04]. Die Idee den Computer zur Verarbeitung natürlicher Sprache zu nutzen ist nicht neu.
Natural Language Processing ist eine Methode zur Verarbeitung und Evaluation der menschlichen Sprache. Der Begriff Natural Language Processing umfasst alle Versuche mit Hilfe eines Computers geschriebene oder gesprochene Sprache für einen praktischen und sinnvollen Nutzen zu verarbeiten.
Natural Language Processing setzt die Einsichten, die durch den Einsatz verschiedener linguistischer Werkzeuge auf geschriebene oder gesprochene Sprache erlangt werden, zur Spracherkennung und zur Sprachsynthese ein:

- Spracherkennung bezeichnet das Erkennen gesprochener Sprache [WAR97],
- Sprachsynthese bezeichnet die Erzeugung von natürlich-klingender Sprache durch einen Computer [STO04].

Eine Auswahl an solchen linguistischen Mitteln findet sich in Kapitel 2.

Die menschliche Sprache dient als Kommunikationsmedium. Sie ist in ihren Ausdrucksmöglichkeiten sehr mächtig.
Eine formale Sprache (z.B. eine Programmiersprache) ist in ihrer Ausdrucksmöglichkeit eingeschränkt. Sie ist immer eindeutig definiert. Die natürliche Sprache dagegen unterliegt dieser Einschränkung nicht. Die Programmiersprache C ist eine solche formale Sprache [KER88]. Die jeweiligen Programme in C stellen die Wörter dieser Sprache dar. Das dazugehörige Alphabet wird durch die per Definition festgelegten Schlüsselwörter und Zeichen von C gebildet.

Sprachverarbeitungssysteme müssen den besonderen Eigenschaften der natürlichen Sprache gerecht werden. Eines der Probleme, die es zu überwinden gilt ist unter anderem die syntaktische Mehrdeutigkeit. Eine solche Art der Mehrdeutigkeit entsteht zum Beispiel aus der Form „Nominalisierung + Bezug + Substantiv“:

„Das Fühlen der Hand....“ kann folgendes bedeuten:

1. Das Gefühl eines Menschen für seine Hand.
2. Das Gefühl das die Hand empfindet.

Die Herausforderung, der sich NLP stellt, ist also unter anderem mit dem hohen Maß an Ambiguität, der natürlichen Sprache, umzugehen. Um die Aufgabe des maschinellen Sprachverstehens zu lösen, wird eine Zerlegung in Teilaufgaben vorgenommen. Sie orientieren sich an den Abstraktionsebenen der Linguistik- der Lehre von der Ordnung (Phonologie, Morphologie, Syntax), vom Inhalt (Semantik), und vom Gebrauch (Pragmatik, Diskurs).

Jede Systemkonstruktion beruht auf einer Modellierung des Gegenstandsbereichs,
welche nach Marr [MAR82] folgenden Stufen enthalten muss: (1) eine abstrakte berechnungstheoretische (was?, warum?) zur Spezifikation des Ziels, (2) eine algorithmische (wie?) zur Spezifikation von Repräsentationsformalismen und Algorithmen und (3) eine implementierungstechnische zur Realisierung einer abstrakten Maschine (welche konkreten Operationen?), die die Aufgabe löst.

Methoden und Verfahren des Natural Language Processing sind:
Spracherkennung, Sprachanalyse (Parsing), Sprachsynthese, Sprachgenerierung, Indexierung (Lemmatisierung), Übersetzung und Tagging.
Im Folgenden wird insbesondere auf das Part of Speech Tagging und 2 spezielle Parsingmethoden eingegangen. Es geht um Chunk Parsing und Shallow Parsing, beides Methoden, die im Rahmen dieser Arbeit verwendet und untersucht wurden.

2.2 Part of Speech Tagging (POS)

Einer der ersten Schritte, die für maschinelles Textverständnis notwendig sind, ist die korrekte strukturelle Analyse des Satzes.
Bestehorn [BES05] führt in seinem Artikel ein weiteres Beispiel an, mit dem deutlich wird, „dass es sogar gänzlich unmöglich werden kann, einen Text zu verstehen, wenn Mehrdeutigkeit und Kontextabhängigkeiten auftreten:“

„Er schlug den Mann mit dem Stock.“

(Beispielsatz 1.1)

Wird hier jemand mit einem Stock geschlagen oder wird hier jemand geschlagen, der einen Stock hat? Das Problem hier ist, dass erst der Kontext vereindeutigt. Somit muss man in der Syntax mit mehreren Hypothesen leben.

Da wegen der Ambiguität menschlicher Sprache eine Verarbeitung, wie sie bei Computer-Sprachen verwendet wird, auf Grundlage des derzeitigen Forschungsstandes nicht möglich ist, versucht man derzeit, Teilprobleme auf dem Weg zur Lösung des Problems der Sprachverarbeitung zu lösen. Eines dieser Teilprobleme ist die Gewinnung von Strukturinformationen aus dem Text, ohne dass Informationen über den Inhalt des Textes vorhanden sein müssen. Einer der ersten Schritte ist das so genannte ***Part of Speech Tagging:***
Part of Speech Tagging besteht aus mehreren Schritten. Zuerst findet die Tokenisierung statt. Das bedeutet, dass der Text in einzelne Tokens unterteilt wird. Einzelne Tokens sind Wortformen oder Satzzeichen.

Ein Token ist nach [BUß83] eine einzelne sprachliche Äusserung und bezeichnet die im Text vorkommenden Wortformen.

Der Satz: *„Die Frau jagt die Katze.“* enthält zum Beispiel 5 Tokens (oder 6, wenn man den Satzendepunkt als eigenes Token zählt). Sätze werden in Tokens zerlegt. Ein Token kann auch aus mehreren Wörtern bestehen.

Dies kann durch eine Segmentierung mittels eines endlichen Automaten geschehen. Anschließend findet die lexikalische oder morphologische Analyse statt. Für jedes Wort werden alle möglichen, der gegebenfalls mehreren, Wortarten bestimmt. Abschließend muss eine Disambiguierung stattfinden. So wird jedem Token in einem Satz das korrekte Tag, die korrekte Wortart zugeordnet. Die Zuweisung des Tags erfolgt durch eine endliche Menge von Tags, die eine Wortart beschreiben. Diese Menge nennt man ***Tagset***.

Tagsets müssen:

- Tags eindeutig definieren
- jedes Token definieren können
- Konventionen für Zweifelsfälle bereitstellen

Ein weiterer Beispielsatz:

„Buchen sie den Flug.“

(Beispielsatz 1.2)

Ziel ist es, daraus folgende Markierung der Satzteile zu erhalten:
Buchen/VVIMP Sie/PPER den/ART Flug/NN ./$

Erklärung nach Wortarteneinteilung im *STTS (Stuttgart/Tübingen TagSet) [SCH95]:*

VVIMP = Verb, Imperativ, voll

PPER = irreflexives Personalpronomen

NN = normales Nomen

ART = bestimmter Artikel

Eine ausführliche Tabelle des STTS-Tagsets befindet sich im Anhang.

Neben dieser korrekten Variante könnte man den Text aber auch wie folgt taggen:

Buchen/NN sie/PPER den/ART Flug/NN ./$

In der ersten Variante wird das Wort Buchen als als ein Verb im Imperativ interpretiert. Im zweiten Fall ist das Wort als Plural des Nomens Buche interpretiert, was offensichtlich falsch ist. Wird für das aktuelle Wort der Tag gesucht, gibt es nach [HER04] verschiedene Informationsquellen für den Tagger:

Syntakmatische Information: Tags der Wörter im Kontext

Lexikalische Information: Wortart des Wortes im Lexikon

Moderne Tagger nutzen heutzutage eine Kombination von syntagmatischer und lexikalischer Information. Zur Disambiguierung existieren verschiedene Ansätze. Man unterscheidet zwischen regelbasierten und stochastischen Taggern.

2.2.1 Regelbasierte Tagger

Regelbasierte Tagger, wie zum Beispiel der Brill-Tagger [BRI92] [BRI94], der später noch genauer erklärt wird, oder TAGGIT [GRE71], basieren auf Regeln, die Beschränkungen ausdrücken, zum Beispiel, dass ein bestimmtes Tag in einem bestimmten Kontext vorkommen muss beziehungsweise nicht vorkommen darf. (Zum Beispiel kein Relativpronomen am Satzanfang)
Die Konventionen müssen nach [MUE06]:

- eindeutig sein,
- intersubjektiv sein und
- möglichst vollständig sein.

Die Eingabe besteht im Wesentlichen aus zwei Teilen:

- einem Text, der keine Tags enthält und
- einem Tagset.

Dies wir nun auf Beispielsatz 1.2 angewandt. Ein Ausschnitt aus einem fiktiven Wörterbuch könnte wie folgt aussehen:

Wort	POS Tag
Buchen	VVIMP
Buchen	VVINF
Buchen	NN
den	ARTDEF
den	PRELS
den	PDS
Flug	NN
Sie	PPER

Abb. 2.2.1: Mögliches Wörterbuch für Beispielsatz 1.2

Ein Wort kann also durchaus mehrere Bedeutungen im Wörterbuch haben. Ein Suchalgorithmus würde nun für jedes Wort in einem Satz eine solche Liste anlegen und mit allen möglichen Wortarttags füllen. In diesem Beispiel könnte eine solche Tabelle so aussehen:

Buchen	{VVIMP, VVINF, NN}
Sie	{PPER}
den	{ARTDEF, PRELS, PDS}
Flug	{NN}

Abb. 2.2.2: Liste für die Wörter aus Beispielsatz 1.2

Für die Worte „Flug“ und „Sie“ wäre die Suche bereits abgeschlossen, da es jeweils nur ein mögliches Element gibt. Es kann davon ausgegangen werde, dass diese beiden Tags richtig sind. Für die beiden anderen Wörter kommen nun die Regeln zur Disambiguierung zum Tragen. Für das Beispiel werden hier zwei einfache Regeln definiert, die gut passen.

```
INPUT: \dem\ , \den\
  if ((+1 NN) && (-1PPER))
  then removeAll (NON ARTDEF-Tags)
  else remove (ARTDEF-Tag)
```

```
INPUT: \Verb\
  if ((-1 Satzgrenze) && (+1PPER))
  then removeAll (NON VVIMP-Tags)
  else remove (VVIMP-Tag)
```

Abb. 2.2.3: Regeln für Beispielsatz 1.2

Die linke Regel besagt, dass bei dem Wort „den“ geprüft wird, ob dem Wort danach ein Tag mit NN zugeordnet ist, und sich davor ein Wort befindet, das mit PPER getaggt wurde. Wenn diese beiden Bedingungen wahr sind, wird jeder Tag aus der Liste entfernt, bis auf den ART-Tag. Das heißt, der Tagger hat heraus gefunden, ob es sich bei dem Wort um einen Artikel handelt, oder nicht.

Sollte eine der beiden Bedingungen nicht erfüllt sein, kann der ART-Tag aus der Liste entfernt werden, da es sich in diesem Fall ganz sicher nicht um einen Artikel handelt. Diese Art der Regelanwendung wird negative Regelanwendung genannt [BES05]. Die rechte Regel verhält sich ähnlich wie die linke. Hier dürfen Verben im allgemeinen als Eingabe benutzt werden. Die Regel sagt aus, dass Wörter, in deren Tag-Menge ein Verb-Tag vorkommt, die am Anfang eines Satzes stehen und die von einem Personalpronomen gefolgt werden, Verben im Imperativ sind.

Regelbasiertes Tagging weist einige Probleme auf. Das Lexikon ist nie ganz vollständig. Außerdem ist die Interaktion der Regeln schlecht überschaubar.
Ein weiterer Nachteil ist, dass die Benutzung von Regeln einer Sprache (zum Beispiel Englisch) für das Tagging einer anderen Sprache unmöglich ist, so dass für jede Sprache ein komplett neues Regelsystem erstellt werden muss. Da man versuchen wollte diesen enorm materiellen und intellektuellen Aufwand zur Erstellung der Regeln zu vermeiden, wurden die stochastischen Tagger entwickelt.

2.2.2 Stochastische Tagger

Stochastische Tagger, wie zum Beispiel CLAWS [LEE83] oder der TnT-Tagger [BRA00], basieren auf den Wahrscheinlichkeiten von Token/Tag-Abfolgen und wählen abhängig davon das jeweils wahrscheinlichste Tag für ein mehrdeutiges Token aus. Ein Artikel, zum Beispiel, erscheint mit hoher Wahrscheinlichkeit vor einem Nomen und im Gegensatz dazu, eher unwahrscheinlich vor einem Verb.

Es werden dabei drei bedingte Wahrscheinlichkeiten benutzt [CHR02]. Die meisten stochastischen Part of Speech Tagger benutzen die *Markov Annahme* [BES05]*:*

Sei $X = \{X_1, X_2, \ldots, Xt\}$ eine Kette von Zufallsvariablen, die Werte aus einem Wertebereich $S = \{s_1, s_2, \ldots, s_n\}$ annehmen können.
Wenn für die Vorhersage des Wertes der Variable X_{t+1} die ***Markov Eigenschaften***

Lokalität (Beschränkte Sicht/Limited Horizon):

$$P(X_{t+1} = s_k \mid X_1, \ldots, X_t) = P(X_{t+1} = s_k \mid X_t)$$

und

Zeitliche Invarianz (Time Invariance):

$$P(X_{t+1} = s_k \mid X_1, \ldots, X_t) = P(X_2 = s_k \mid X_1)$$

gelten, dann wird X als ***Markov Kette*** bezeichnet.

„Nur der unmittelbar vorhergehende Prätext ist relevant für die Bestimmung. Der Umfang des Kontextes gibt die Ordnung wieder.“

Ein ***Hidden Markov Model*** (HMM) ist ein Fünftupel H = (S, K, π, A, B) mit

S:= $\{s_{1,....,}s_n\}$ Menge aller Zustände

K:= $\{k_{1,....,}k_m\}$ = {1,...,m} Menge der Ausgangssymbole

π:= $\{\pi i\}$ Wahrscheinlichkeiten der Startzustände, sie gibt für jeden Zustand an, mit welcher Wahrscheinlichkeit er den Startzustand bildet (also als X_1 bezeichnet wird):
$\pi_i = P (X_1 = s_i)$

A:= $\{a_{ij}\}$ Zustandsübergangsmatrix, wobei a_{ij} die Wahrscheinlichkeit angibt, dass nach Zustand s_i in Zustand s_j gewechselt wird.

B:= Wahrscheinlichkeiten der Symbolemission in einem Zustand, das heißt, die Wahrscheinlichkeit, dass zur Zeit t das Symbol x beobachtet wird, unter der Vorraussetzung, dass das Modell sich zur Zeit t sich im Zustand S_i befindet und zum Zeitpunkt t+1 in den Zustand S_j übergeht.
$b_i(x)$ bezeichnet demnach die Wahrscheinlichkeit, im Zustand s_i die Beobachtung x zu machen.

Die Summe der Wahrscheinlichkeiten der

- Startzustände,
- der ausgehenden Kanten von einem Zustand und
- der Symbolemissionen in einem Zustand

müssen jeweils genau 1 sein [MAN00].

Für betrachtete n-Gramme mit n = 3 haben wir es mit einem Markov-Modell 2.Ordnung zu tun (n-1.Ordnung) .

Man kann nach [HER04] sagen, dass stochastische Tagger dieser Art folgende Annahmen treffen:

- Kontextbegrenzung: Ein Wort ist nur abhängig von dem Tag des vorhergehenden Wortes. Es hat sich gezeigt, dass diese Annahme relativ gute Ergebnisse (-90% korrekt markierte Wörter)erzielt.
- Zeitinvarianz: Die Abhängigkeit bleibt innerhalb des Modells zu jedem Zeitpunkt konstant.
- Die Wörter sind unabhängig voneinander.
- Die Wortidentität hängt nur von seinem Tag ab.

Allerdings lässt sich für jede dieser Annahmen ein natürlich-sprachliches Phänomen finden, welches dieser Annahme widerspricht.
Es ergibt sich dadurch das Problem der ***Sparse Data*** („Problem der knappen Daten“) :

> „Die größte Zahl der Kollokationen in N kommt nur ein bzw. sehr wenige Male vor => geringe Ausbeute an möglichen Gebrauchskontexten.“
>
> Der Begriff Kollokation steht in diesem Zusammenhang für das benachbarte Auftreten von zwei Wörtern (Wortformen). N bezeichnet die Anzahl der N-Gramme (Tokens). Das Problem der Datenspärlichkeit lässt sich auch nicht durch größere Trainingskorpora vermeiden [TRE06].

Für jede mögliche Wortsequenz der Länge n über einem Vokabular w wird die Wahrscheinlichkeit

$P(w_i \mid w_{i+n+1}, \ldots w_{i-1})$

mit einer ausgewählten Methode geschätzt. Eine Methode ist die ***Maximum Likelihood Estimation*** (MLE):

Die Wahrscheinlichkeit eines *N-Gramms* ist gleich der relativen Häufigkeit im Trainingskorpus:

$P(w_{1, \ldots, w_n}) = C(w_{1, \ldots, w_n}) / N_{n\text{-Gramme}}$ (C = Anzahl der N-Gramme)

Die *bedingte Wahrscheinlichkeit* für w_n gegeben den Kontext $w_1,\ldots, w_{n-1}$ ist

$P(w_n \mid w_1,\ldots, w_{n-1}) = C(w_{1, \ldots, w_n}) / C(w_1,\ldots, w_{n-1})$.

Für nicht-trainierte Vorkommen, das heißt prinzipiell mögliche, aber im Trainingskorpus nicht vorkommende N-Gramme, macht die *Maximum Likelihood Estimation* eine Voraussage für P = 0, wodurch es zu einer Notwendigkeit anderer Schätzer kommt.
Es wird eine Wissensbasis erstellt, damit ein Tagger weiß, welches Wort in welchem Zusammenhang eine bestimmte Wortart hat. Dazu werden große, von Hand annotierte Korpora genutzt, die von Hand mit dem grammatikalisch korrektem Part of Speech Tag versehen wurden. Zwei große bekannte Korpora sollen hier als Beispiele erwähnt werden:

- Negra: Korpus mit mehr als zwanzigtausend Sätzen (circa 350000 Wörter in deutscher Sprache der Frankfurter Rundschau)
- Brown Korpus: Mehr als 1 Million Wörter aus circa 500 englischen Texten aus verschiedenen Bereichen (z.B. Zeitungen, wissenschaftliche Arbeiten, Anleitungen...) [PTT93]

Mit Hilfe der Matrizen soll jetzt für die Beobachtungen ein Tagging erzeugt werden, welches die größte mögliche Wahrscheinlichkeit hat.

Dazu wird zunächst das ***Bayes Theorem*** definiert. Mit seiner Hilfe lassen sich unbekannte Parameter schätzen. Für zwei Ereignisse A und B lautet es:

$$P(A|B) = P(B|A) * P(A)/P(B)$$

Wobei

P(A) die A-Priori-Wahrscheinlichkeit für Ereignis A ist,

P(B|A) die Wahrscheinlichkeit für Ereignis B unter der Bedingung, dass A eingetreten ist und

P(B) die A-Priori-Wahrscheinlichkeit für Ereignis B ist.

Phase 1: Initialisierung

Bei der Initialisierung wird nur ein Startknoten S erzeugt. Die Funktion $\delta(i,t)$ berechnet die Wahrscheinlichkeit, dass das i-te Wort mit dem Tag t versehen wird. Dem Startknoten wird $\delta(0,START)=1$ zugewiesen. Die Funktion $\psi(i,t)$ berechnet für das i-te Wort mit dem Tag t den Vorgängerknoten.

Phase 2: Aufbau des Graphen

Hier werden für jedes Wort *i* und jeden Tag im Tagset *t* die beiden Funktionen $\delta(i,t)$ und $\psi(i,t)$ berechnet.

Für den Satz "*Buchen Sie den Flug"* wird damit der Graph wie folgt aufgebaut:

Nachdem durch den Initialisierungsschritt der Knoten S erzeugt wurde und $\delta(0,START) = 0$ gesetzt wurde, wird für jeden Tag *t* ein Knoten ("Buchen",t) erzeugt.

Am Beispiel des Knotens (“Buchen”,NN) sollen die nachfolgenden Schritte verdeutlicht werden:

1. Es wird $\psi(1,NN) = S$ gesetzt, das heißt der Vorgängerknoten von (“Buchen”,NN) ist S.

2. Um den Wert der Wahrscheinlichkeitsfunktion δ zu berechnen werden die folgenden Werte aus den Matrizen benötigt:

a) Der Wert $P(NN|Buchen) = b11 = 0,2$, der die Wahrscheinlichkeit angibt, mit der das Wort “*Buchen”* mit NN markiert wird. (Ereignis A)

b) Der Wert $P(START|NN) = \Pi 1 = 0,2$, der die Wahrscheinlichkeit angibt, mit der ein Nomen (NN) am Anfang eines Satzes steht. (Ereignis B)

Ist P(A) = P(A|*B)* sagt man, dass zwei Ereignisse unabhängig voneinander sind. Die gegebenen Ereignisse sind demnach unabhängig voneinander. Für die Wahrscheinlichkeit P(A∩B), also die Wahrscheinlichkeit, dass beide Ereignisse gemeinsam auftreten, muss das Produkt der Einzelwahrscheinlichkeiten gebildet werden. Durch die Multiplikation der beiden Werte $\Pi 1 \leftrightarrow b11 = 0,04$ kann nun der endgültige Funktionswert von $\delta(1,NN)$ berechnet werden. Dieser ergibt sich aus dem Produkt des eben berechneten Wertes und dem Wert von $\delta(0,\psi(1,NN)) = 1,0$, also dem δ-Wert des Vorgängerknotens.

Die vorgestellten Schritte werden für jeden Tag t durchgeführt, wobei die Knoten, deren δ-Wert gleich 0 ist, nicht weiter bearbeitet werden müssen. Der aufgebaute Graph sieht so aus:

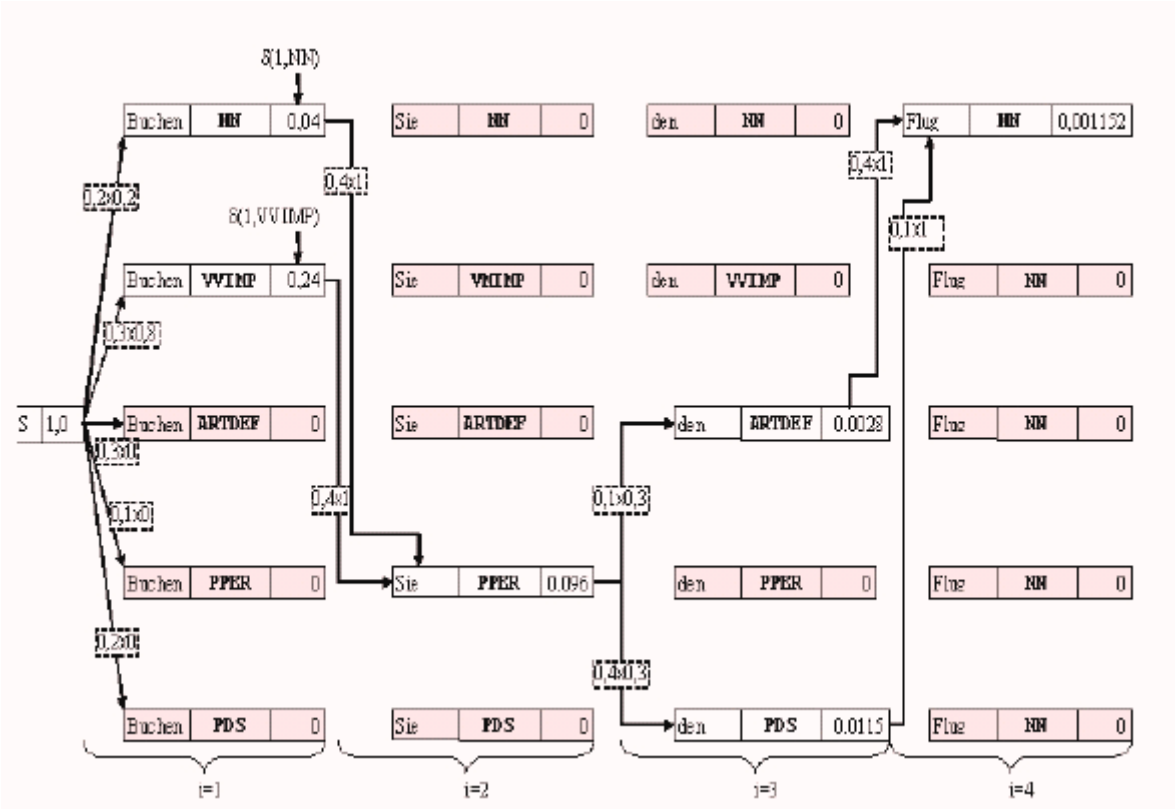


Abb. 2.2.4. Aufgebauter Graph für Beispielsatz 1.2

Aus diesem aufgebauten Graphen kann man das wahrscheinlichste Tagging ablesen. Seine Knoten enthalten die Wahrscheinlichkeiten eines Taggings, das durch den Pfad vom Startknoten zu diesem Knoten repräsentiert wird. Jeder der dargestellten Knoten entspricht einem Wort und dem dazugehörigen Tag. Am Ende gibt es zwei Knoten, deren δ-Wert ungleich 0 ist, nämlich (“Buchen”,NN) und (“Buchen”,VVIMP). Diese beiden Knoten sind nun die einzigen beiden Knoten, die weiter bearbeitet werden müssen.

Am Beispiel des Knotens („Sie”,PPER) wird nun gezeigt, wie der Graph weiter aufgebaut wird, bis alle Knoten expandiert sind:

1. Es gibt zwei Möglichkeiten zum Endknoten zu gelangen, die eine ist der Pfad über den Knoten (“Buchen”,NN), die andere Variante wäre über den Knoten (“Buchen”, VVIMP). Es müssen also die δ-Werte für beide Varianten errechnet werden:

a) (“Buchen”,NN): Die Wahrscheinlichkeit *P(Sie|PPER) = b42 = 1* wird multipliziert mit *P(PPER|NN) = a13 = 0,4*, was einen Wert von 0,4 ergibt. Dieser Wert wird nun mit δ*(1,NN) = 0,04* multipliziert, was 0,016 ergibt.

b) (“Buchen”,VVIMP): Hier wird analog zu oben verfahren, indem zunächst das Produkt *P(Sie|PPER) ↔ P(PPER|VVIMP) = b42↔ a23 = 0,4* gebildet wird. Das Ganze wird nun mit dem δ-Wert des Vorgängerknotens 0,24 multipliziert, womit δ*(2,VVIMP)= 0,096* gilt.

2. Die anderen Vorgänger-Knoten müssen nicht miteinbezogen werden, da hier sich eine Multiplikation mit 0 ergeben würde.

3. Aus den beiden Varianten wird nun diejenige ausgewählt, die die größere Wahrscheinlichkeit hat, also letztere. Für den Knoten (“Sie”,PPER) gilt damit δ*(2,PPER) = 0,096* und ψ*(2,PPER)* =(“Buchen”, VVIMP). Wie beschrieben werden die restlichen Knoten für die restlichen Wörter aufgebaut. Wenn der Graph fertig ist, kann in der dritten Phase des Algorithmus ein Pfad ausgelesen werden, der das Tagging mit der größten Wahrscheinlichkeit repräsentiert.

Phase 3: Auslesen des Part of Speech Taggings

Aus den Knoten des letzten Schrittes beim Aufbau des Graphen, also im Beispiel mit *i=4*, wird nun derjenige ausgewählt, der die größte Wahrscheinlichkeit, also den größten δ-Wert hat. Für das Beispiel wäre das der Knoten (“Flug”,NN) mit δ(4,NN) = 0,001152. Von diesem Knoten aus kann nun solange mit Hilfe der ψ-Funktion immer der Vorgänger-Knoten ermittelt werden, bis der Startknoten S erreicht ist. Für jeden Knoten werden also die folgenden Schritte durchgeführt:

1. Berechne den Vorgänger des aktuellen Knotens mit der ψ-Funktion.

2. Speichere den berechneten Knoten in einer FIFO-Warteschlange (First in – First out) ab.

Wie zu erwarten, ergibt sich dadurch das gewünschte Tagging des Beispielsatzes 1.2:

“Buchen Sie den Flug”

Buchen/VVIMP Sie/PPER den/ART Flug/NN

(Der Punkt am Satzende wird hier bewusst weggelassen.)
Im Gegensatz zu den regelbasierten Taggern fällt das Aufstellen komplexer Regelsysteme zur Elimination Doppeldeutigkeiten bei stochastischen Taggern weg. Aber auch beim stochastische Tagging gibt es Probleme [MUE06]. Eine Wahrscheinlichkeit bedeutet nicht immer Sicherheit. Der Tagger ist sehr abhängig vom Trainingskorpus. Neben den beiden bereits erwähnten Methoden des Taggens gibt es noch einen weiteren bekannten Tagger, den Brill-Tagger., der auf *Transformation Based Learning (TBL)* basiert.

2.2.3 Der Brill Tagger

Der 1992 von Eric Brill entwickelte Brill Tagger [BRI92] [BRI94] basiert auf der Kombination von regelbasierten und stochastischen Verfahren und einem getaggten Korpus. Er betrachtet nur das zu taggende Wort und dessen nahe Umgebung um zu entscheiden, welche Wortklasse am wahrscheinlichsten ist. Die Regeln und statistischen Werte lernt der Tagger selbstständig aus einem manuell korrigierten Trainingskorpus. Zur Ermittlung des "besten" Tags für ein zu taggendes Wort geht der Tagger wie folgt vor. Zuerst teilt der Tagger jedem zu taggenden Wort ein provisorisches Tag zu, dann versucht er anhand von Regeln, die Wortklasse zu präzisieren, indem er Tags von bestimmten Kandidaten unter gewissen Bedingungen in andere transformiert. Der Weg vom ersten provisorischen bis zu endgültigen Tag kann mehrere solche Transformationen durchlaufen.

Ein Brill-Tagger besteht aus mehreren Komponenten:

- Lexikon: Es besteht aus einer Liste von Wörtern, mit ihren möglichen Tags, wobei das im Trainingskorpus wahrscheinlichste Tag zuerst steht.
- Transformation Rules (TR): Diese Regeln werden zur Korrektur von Tagging-Fehlern genutzt. Durch Nutzung des Transformation Based Learning können die Regeln „automatisch“ gelernt werden und müssen nicht manuell definiert werden. In Zusammenhang mit Beispielsatz 1.2 könnte dies wie folgt aussehen:

  „Buchen/VVIMP Sie/PPER den/ART Flug/NN.“
  „Die/ART Buchen/VVIMP sind/VVAIMP groß/ADJ.“

  Das Tagging des Wortes Buchen im 2.Satz ist offensichtlich falsch, da es sich hier nicht um einen Imperativ handelt. Entstandene Fehler können durch den Vergleich mit dem manuell annotierten Trainingskorpus gefunden werden und werden anschließend in einer Tabelle, die die Fehler kategorisiert festgehalten. [BES05].

Die Tabelle in Abbildung 2.2.5 enthält Tripel {t_{Haben}, t_{Soll}, Anzahl}, in denen die Anzahl der Fenster gespeichert werden, bei denen ein Tag t_{Haben} statt t_{Soll} verwendet wurde:

t_{Haben}	t_{Soll}	Anzahl
VVFIN	NN	134
VVIMP	NN	1
ADJ	VVFIN	54

Abb. 2.2.5. Tabelle zur Fehlerdarstellung

Die Tabelle zeigt, dass es an 134 Stellen den Fehler gab, dass ein Wort, welches korrekt markiert ein Nomen ist, fälschlicherweise als finites Vollverb markiert wurde.

„Der Brill Tagger versucht, aus diesen Fehlern zu lernen und automatisch Regeln zu erzeugen."

- Lexikalische Regeln (LR): Präfix- und Suffix-Wahrscheinlichkeiten werden aus einem Trainingskorpus automatisch ermittelt. Mit Hilfe von morphologischen Regeln wird versucht Wörtern, die nicht im Lexikon vorhanden sind, den richtigen Tag zuzuweisen. Die selbst gelernte Regel:

  bar hassuf 3 ADJD 5

  besagt, dass ein Wort mit dem 3-buchstabigem Suffix *-bar,* mit einem Wahrscheinlichkeitswert von 5, ein Adjektiv in prädikativer Position (ADJD) ist.

- Bigrammregeln: Für nicht im Lexikon gefundene Wörter zieht der Tagger eine Sammlung von Bigrammen zu Rate. Diese Sammlung wird aus einem beliebig grossen, in der Art dem Trainingskorpus verwandten, aber nicht getaggten Korpus gewonnen. Aus dieser Bigrammsammlung findet der Tagger heraus, welche Wörter häufig unmittelbar vor oder hinter dem Kandidaten zu pflegen stehen (*adjazente Paare*). Je nach deren Wortart entschliesst sich der Tagger dazu, das provisorische Tag des Kandidaten zu belassen oder in eine anderes zu transformieren.

- Kontextregeln: Kontextregeln gelten für bekannte Wörter und können über den Bigrammrahmen hinausgreifen. Sie transformieren ein provisorisch dem Wort zugewiesenes Tag in ein anderes im Lexikon aufgeführtes, falls es aufgrund der Wortumgebung angebracht ist. Die Regel

  VVFIN VAFIN NEXT1OR2OR3TAG VVPP

  besagt, dass flektierte Vollverben (VVFIN) in flektierte Hilfsverben (VAFIN) transformiert werden sollen, falls in den nächsten 3 Wörtern ein Partizip (VVPP) auftaucht. „Diese Regel kommt bei Verben zum Einsatz, die gemäss Lexikon sowohl Voll- als auch Hilfsverben sein können, also vor allem bei *sein* und *haben*."

Zu Beginn weist der Tagger jedem bekannten Wort ein initiales Tag zu. Anschließend werden den Wörtern, die nicht im Trainingskorpus vorkamen, Tags mit Hilfe von Kontextregeln und Bigrammen zugewiesen. Abschließend werden zur Korrektur von Fehlern, die in den ersten beiden Schritten gemacht wurden, iterativ die lexikalischen Regeln angewandt, bis nichts mehr zu verbessern ist. Angewandt auf Beispielsatz 1.2 bedeutet dies:

Die Menge der Beobachtungen ist:

O={ O1= Buchen, O2= Sie, O3= den, O4= Flug}

Ein Vorteil gegenüber den stochastische Taggern ist, dass viel weniger Informationen gespeichert werden müssen. Die Kontextinformationen müssen bei stochastische Taggern in sehr großen Matrizen gespeichert werden. Der Brill Tagger kommt mit weniger als 100 Regeln aus. Er lernt die Regeln selbstständig. Da der Brill Tagger ein eigenes Lexikon aufbaut, kommt er ohne externes Lexikon aus.

In der kontextuellen Phase des Taggers gibt es Nachteile [AND05]. Jede Regel wird im Input Wort für Wort angewendet. Dadurch werden viele Wörter mehrmals hintereinander geprüft. Alle Regeln werden der Reihe nach an jeden Satz angewendet, dadurch können die Regeln miteinander interagieren. Es kommt zu einer großen Verlangsamung aufgrund dieser unnötigen Operationen.

2.3 Parser

Nachdem das Tagging abgeschlossen ist widmet man sich dem Parsen der Sätze. Ein anderes Wort für Parsen ist Syntaxanalyse. Aufbauend auf der besten (oder auch den n-besten) POS-Tag-Kette(n) zu einem Satz wird die syntaktische Analyse durchgeführt. Ein komplexer Ausdruck wird automatisch in seine Konstituenten zerlegt. Der Text ist so für eine weitere Verarbeitung aufbereitet.
Parsen natürlicher Sprache birgt nach [SCH02] einige Hürden, die es bestmöglich zu überwinden gilt:

- Komplexität (Strukturvielfalt),
- Vagheit und
- Ambiguität (syntaktische, lexikalische, referentielle Mehrdeutigkeiten).

Abbildung 2.3.1 zeigt die Einordnung des Parsers in den Prozess der Sprachverarbeitung nach [SCH02].

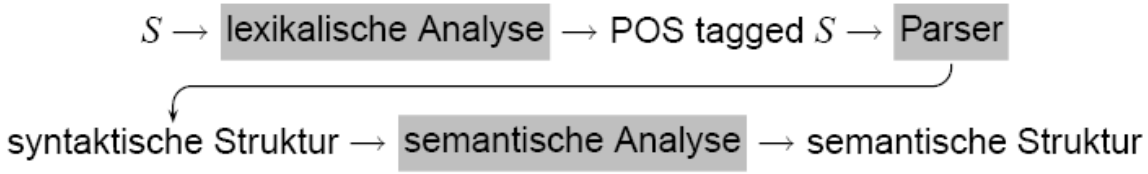


Abb. 2.3.1: Rolle des Parsers im Prozess der Sprachverarbeitung

Ein Parser besteht aus einer Menge von Syntaxregeln und dem Parsingalgorithmus. Abhängig von der Architektur kann ein Parser Zugriff auf ein Lexikon haben, oder das Lexikon ist in den Parser integriert.

Es gibt unterschiedliche Methoden und Strategien zu parsen.

- Analyserichtung: top-down vs. bottom-up,
- Suchstrategien: breadth-first (parallel) vs. depth-first (sequentiell) vs. Best-first (heuristisch),
- Verfahrensweise: deterministisch vs. nicht-deterministisch,
- Verarbeitungsrichtung: uni- vs. Bidirektional,
- Kommunikation: online vs. offline,

Im Kontext dieser Arbeit ist die Unterscheidung zwischen vollständigem und partiellem Parsing sinnvoll. Die Systeme unterscheiden sich im Grad der Tiefe der syntaktischen Analyse. Aufgrund der großen Anzahl an verbreiteten Parsern ist es nicht möglich im Rahmen dieser Arbeit alle Parser umfassend zu beschreiben. Vielmehr stellt sich die Aufgabe eine geeignete Analysetiefe für die sehr fehlerbehafteten Texte zu finden.

2.3.1 Vollständiges Parsing

Beim vollständigen Parsing ist eine vollständige und korrekte Analyse das Ziel. Die Grammatik ist vollständig und gesucht wird die beste Lösung des gesamten Suchraumes. Man erreicht mit dieser Methode gegenüber dem partiellen Parsen eine bessere Übereinstimmung mit linguistischen Theorien. Die Syntaxregeln sind jedoch komplexer. Ein Vorgehen nach diesem Lösungsansatz bietet sich an, wenn man es zum Beispiel in einer recht begrenzten Anwendungsdomäne, mit einfachen wohlgeformten Sätzen arbeitet. Man kann dann erwarten, dass die Syntaxregeln vergleichsweise gering sind, sodass man mit wenig Problemen bei der Grammatikbeschreibung rechnen kann. Andernfalls muss man bei der Tiefenstrukturanalyse überwiegend von einem komplexeren Grammatikformalismus ausgehen.
Durch den Vollständigkeitsanspruch steigt der Berechnungs- und Speicheraufwand exponentiell an. Die Effizienz dieses Verfahrens nimmt also ab. Das liegt unter anderem daran, dass rekursive Strukturen und sprachliche Phänomene, wie Fernabhängigkeiten berücksichtigt werden.
Ein Beispiel dafür ist HPSG. HPSG steht für Head-driven Phrase Structure Grammar. HPSG wurde zuerst beschrieben in [POL87] später überarbeitet und in [POL94] neu beschrieben.

Die HPSG nutzt zur Modellierung linguistischer Objekte sogenannte Merkmalsstrukturen. Ursula Klenk beschreibt die HPSG so [KLE03]:

> „Syntax und Semantik von Ausdrücken werden integriert in einer einzigen Merkmalsstruktur dargestellt. Die Semantik der HPSG orientiert sich an der Prädikatenlogik erweitert um das Konzept semantischer Rollen von Argumenten und Behandlung spezieller natürlich-sprachlicher Gegebenheiten, wie z.B. Semantik von Artikelwörtern und Quantifizierern. Dabei werden natürlich-sprachliche nicht nur in logiksprachliche übersetzt, sondern ihren syntaktischen Merkmalsstrukturen semantische Merkmalsstrukturen zugeordnet, d.h. Es erfolgt eine Übersetzung der Ausdrücke zusammen mit der Beschreibung des Strukturaufbaus dieser Übersetzungen. Die integrierte Darstellung von Syntax und Semantik erlaubt es, beide Komponenten durch Verweise in Beziehung zueinander zu setzen [„„]“

Verfahren, die ein vollständiges Parsing zum Ziel haben, haben die „ganz oder gar nicht“-Eigenschaft. Entweder die vollständige Analyse gelingt komplett, oder gar nicht.. Typischerweise enthält die Eingabe wohlgeformte Konstituenten, die nicht geparst werden können/für die keine Grammatikregeln existieren, was zur Folge hat, dass der Parser fehlschlägt. Realistische, natürlichsprachliche Sätze sind außerdem meist durch Schreibfehler, Auslassungen und ähnlichem fehlerbehaftet. Speziell im Fall der Schulkinderaufsätze kommt dies enorm zum Tragen. Mit einem vollständigen Parsing, würde man nur sehr wenige Ergebnisse erzielen können. Deshalb bietet sich die Möglichkeit des partiellen Parsings an.

2.3.2 Partielles Parsing

Ziel dieser Art zu parsen ist, durch die flache- beziehungsweise Breitenstrukturanalyse eine breite Datenabdeckung zu erreichen. Es wird nicht der gesamte Input analysiert, es werden nur bestimmte Teilstrukturen eines Satzes erkannt. Diese Konstituenten besitzen einen einfachen Aufbau und sind deshalb leicht, schnell, zum Beispiel mit Hilfe endlicher Automaten und mit großer Zuverlässigkeit identifizierbar:

- Kernnominalphrasen (z.B. „dieses sehr schöne Haus“)
- Präpositionalphrasen (z.B. „mit vielen Fenstern“)
- Verbkomplexe (z.B. „kaufen hat müssen“)

Nach [Frederici *et al.,* 1996] ist diese Art von Analyse also durch zwei Merkmale charakterisierbar:

> „Das Ergebnis eines flachen und partiellen Parsers muss keine vollständige Analyse der Eingabe sein. Es kann syntaktisch unterspezifiziert bleiben. Die Auflösung solcher Unterspezifikationen findet – wenn überhaupt – in einem späteren Verarbeitungsschritt statt.
>
> Ein solcher Parser arbeitet auf einem Minimum an linguistischem Wissen. Sein Vorgehen ist nicht lexikalisch gesteuert.“

Diese Art von Analyse ist robust, aber weniger korrekt. Man arbeitet mit einer einfachen Grammatik. Die Syntaxregeln sind geringer. Geeignet ist diese Methode insbesondere für ungeformte und große Satzstrukturen. Betrachtet man die Aufsätze, die die Schulkinder geschrieben habe, erkennt man, dass partielles Parsing der richtige Lösungsansatz ist. Ein flacher Parser wird auch Shallow Parser genannt und ist ein Parser, der die Struktur eines Satzes nicht in jedem Fall bis auf die Ebene von präterminalen Elementen ermittelt.
Eine spezielle Art des partiellen Parsings ist das ***Chunk Parsing***.

2.3.2.1 Chunk Parsing

Der Chunk Parser ist auf Robustheit und Effektivität ausgelegt.
„Es ist ein unvollständiges, nicht-rekursives und lokales Verfahren.“ [HES06]
Sätze werden in sinnvolle Wortgruppen (Konstituenten) zerlegt. Bestimmte Abfolgen von Wortformen werden zu einem „Chunk“ zusammengefasst. Steven Abney [ABN91] definiert einen Chunk als ein Informationssegment in einem Satz. Ein Chunk ist eine Sequenz von Wörtern, gelabelt mit einer syntaktischen Kategorie .

> „I define chunks in terms of major heads. Major heads are all content words except those that appear between a function word f and the content word that f selects.”
> (vgl. Abney 1991)

Chunks sind Teile sogenannter Performanzstrukturen. Die Existenz solcher Strukturen wurde von [Gee und Grosjean, 1983] nachgewiesen. Das Auftreten von Performanzphänomenen (Abbrüche, Korrekturen, Häsitationen[1]) spielt bei der Verarbeitung spontan gesprochener Sprache eine große Rolle. Dies bringt „erhebliche Abweichungen vom Ideal wohlgeformter sprachlicher Äußerungen mit sich.“
Chunks werden auf rein syntaktischer Ebene definiert. Semantische und funktionale Faktoren spielen keine Rolle.

Wenn man einen Satz liest, liest man einen Chunk nach dem anderen [ABN91], Abney führt dazu folgendes Beispiel an:

> [I begin] [with an intuition]: [when i read] [a sentence], [i read it] [a chunk] [at a time]

Das Ziel ist also, einen Satz in Stücke, in eine Sequenz von Chunks zu zerlegen. Chunks sind nicht überlappende Bereiche eines Satzes. Sie sind nicht rekursiv, das heißt ein Chunk enthält keinen weiteren Chunk. Außerdem sind sie nicht abdeckend, das heißt je nach Chunk-Definition sind manche Wörter eines Satzes nicht Teil eines Chunks:

> [take] [the second road] that [is] on [the left hand side]

1 Häsitationen sind Artikulationen, die den flüssigen Wortstrom unterbrechen (z.B. ähm, öh, hm) [MEN06]

Chunks haben meist einen Kopf, auch *Head* genannt, der durch die anderen Elemente modifiziert wird:

[walk] [straight **past]** [the **lake]**

Chunks sind üblicherweise Subsequenzen von Konstituenten, das heißt es gibt keine Überschneidungen mit den üblichen Konstituentengrenzen.
Die Anzahl der Lösungen durch Chunking ist viel geringer als bei einem System, das vollständiges Parsing nutzt, wie zum Beispiel FASTUS [HOB96].

Chunking kann mit endlichen Automaten, also mit regulären Ausdrücken realisiert werden. Üblicherweise verarbeitet ein Parser für natürliche Sprache einen Text in zwei Schritten. Ein Tokenizer, beziehungsweise eine morphologische Analyse wandelt eine Reihe von Buchstaben, in eine Reihe von Wörtern um, und der Parser wandelt eine Reihe von Wörtern in einen oder eine Reihe von geparsten Sätzen um [ABN91].

In einem Chunk Parser ist der syntaktische Analyseprozess in zwei Stufen unterteilt. Abney nennt diese Stufen den Chunker und den Attacher. Der Chunker konvertiert Wortketten zu Chunks und der Attacher konvertiert Ketten aus Chunks zu Sätzen. Chunker und Attacher sind nicht-deterministische LR-Parser mit ähnlichem Aufbau. Ein LR-Parser ist ein deterministischer bottom-up Parser. Der Parser schiebt Wörter aus der Eingabe auf den Stack, den Stapel, zu dem ein Zugang nur von oben möglich ist, bis er eine Abfolge von Wörtern erkennt, die mit der rechten Seite einer Grammatikregel übereinstimmen. Dann reduziert er die Sequenz bis auf einen einzelnen Knoten, dessen Kategorie auf der linken Seite der Regel gegeben ist [AHO06]. Es werden neue Knoten in den Syntaxbaum eingefügt, um die Subgraphen zu komplettieren. Zusätzlich werden neue Kanten eingefügt, um die Subgraphen miteinander zu verbinden. Die Hauptaufgabe des Attachers ist der Umgang mit den Mehrdeutigkeiten, die bei dem Anhängen entstehen.

Die Attachmentstellen sind nach Priorität folgendermaßen einzuordnen:

1. Attachment als Verbargument (zum Beispiel Subjekt, Objekt)
2. Attachment als Argument eines Nicht-Verbs (zum Beispiel Attribute)
3. Attachment als Verbmodifizierer (zum Beispiel Adverbien, Modaladverben)
4. Attachment als Modifizierer eines Nicht-Verbs (zum Beispiel Modalpartikel/Füllwörter)

Abney sagt außerdem, dass niedriges Anhängen im Syntaxbaum zu bevorzugen ist. Hier ist die relative Höhe der Attachmentstelle im Baum gemeint. Jedem gegebenen Wort wird ein Frameset zugeschrieben, das die obligatorischen und die optionalen Argumente des Wortes in Form von Slots darstellt (<:“only appears first“, >:“only appears last“) (Vgl. Abney 1991).
Steven Abney nennt drei Schlüsselkonzepte, mit denen sich diese Parsingstrategie charakterisieren lässt [ABN96]:

- **Easy-first parsing:** Es werden nur die Segmente im Text markiert, die sich unter dem Einsatz der lokalen Information relativ eindeutig erkennen lassen.
- **Islands of certainity:** Jede Stufe setzt die Ergebnisse der vorhergehenden Stufen zu immer größer werdenden, sicher erkannten Strukturen zusammen.
- **Containment of ambiguity:** Mehrdeutigkeiten werden nicht unbedingt aufgelöst, aber eingeschränkt.

Durch diesen Aufbau hat der Chunk Parser einige Vorteile zu bieten. Syntaktische Attachment-Ambiguitäten tauchen innerhalb von Chunks nicht auf, da nicht Wörter sondern ganze Chunks als Einheit geparst werden. Semantische Ambiguitäten werden im Attacher bearbeitet. Die aufwendigen Techniken bleiben also auf den Attacher beschränkt. Da der Chunker unempfindlich gegenüber dem Zustand des Attachers ist kann man ihn auch separat vom Attacher entwickeln und debuggen.

2.4 Evaluation

In diesem Kapitel wird eine genaue Beschreibung der Bedeutung von Evaluation, wie sie im Kontext dieser Arbeit genutzt und verstanden wird gegeben. Es wird speziell auf die Evaluation von Parsern eingegangen. Die verwendeten Erfolgs- und Qualitätsmerkmale, die für die Evaluationsstudie von Bedeutung waren und genutzt wurden, werden definiert und erläutert. Außerdem werden einige, bereits praktizierte und verbreitete Verfahren für die Evaluation von Parsern kurz vorgestellt und beschrieben.

2.4.1 Allgemeine Betrachtungen

Vorab wird eine Beschreibung von Evaluation im Allgemeinen gegeben. Anschließend wird speziell auf die Evaluation von Parsern eingegangen. Peter Baumgärtner stellt folgende These über die Evaluation auf [BAU99]:

> „Unter Evaluation sind alle Aktivitäten und/oder Ergebnisse zu verstehen, die die Bedeutung, Verwendbarkeit, (Geld-) Wert, Wichtigkeit, Zweckmäßigkeit, ... einer Sache beurteilen bzw. bewerten. Nur dieses weit gefasste Verständnis von Evaluation kann sowohl die Charakteristika besonderer Evaluationsfelder berücksichtigen als auch einen adäquaten Beitrag zur Theoriebildung leisten."

Anders als Grundlagenforschung orientiert sich Evaluation an den konkreten Fragen von Entscheidungsträgern. Damit ist sie auf ein breites Spektrum an Methoden angewiesen, um (zeitgerecht) hilfreiche Information bereit zu stellen. Evaluation befasst sich mit Handlungsfeldern, in denen herkömmliche ökonomische Bewertungen zu kurz greifen.

Nach dem internationalen Standard DIN ISO 9162 teilen sich die Softwarequalitätsmerkmale in sechs Gruppen:

- Funktionalität,
- Zuverlässigkeit,
- Benutzbarkeit,
- Effizienz,
- Änderbarkeit und
- Übertragbarkeit.

Eine Evaluation hat zielorientierte, sachbezogene Ziele, vorliegende Ist-Werte werden mit angestrebten und/oder erwarteten Soll-Werten verglichen. Sie dient zugleich der rückblickenden Wirkungskontrolle wie auch der vorausschauenden Steuerung.

> „Evaluation ist die systematische Untersuchung von Qualität oder Nutzen einer Sache“ [Joint committee on standards for educational evaluation, 1994]

Vergleichbare und reproduzierbare Evaluationen und Experimente sind wichtige Quellen für die Forschung.

Gerade in der vorliegenden Problemstellung ist die Evaluation des untersuchten Systems, also SMES, gegenüber einem fehlerhaften Input interessant.

2.4.2 Evaluation von Parsern

Eine gültige und vergleichbare Evaluation von Natural Language Processing-Systemen ist schwer zu erreichen, ohne eine beträchtliche Menge „händischer“ Arbeit. Doch „händische“ Arbeit ist kosten- und zeitintensiv und nicht unbedingt fehlerfrei. Es gab bereits mehrere verschiedene Ansätze für die Evaluation von Parsern im Laufe der Jahre [LIN95] [MOL03] [BIG05] [BIG03].

Je nach Art einer Anwendung kann ein Text nahezu fehlerfrei sein (zum Beispiel Zeitungsartikel), oder er kann eine hohe Anzahl an Fehlern enthalten, wie zum Beispiel bei der Textsorte „gesprochene Sprache“ oder wie es bei den Texten der Schulkinder zu erwarten ist. Es ist nicht ausreichend, die Performanz eines Parsers in Bezug auf einen fehlerfreien Textkorpus zu optimieren, viel wichtiger ist es, die Performanz eines Parsers in Bezug auf einen fehlerbehafteten zu optimieren. Es geht darum die Robustheit eines Parsers zu evaluieren. Robustheit ist in diesem Kontext als die Robustheit gegenüber schlecht-geformten Input. Der Parser sollte so robust sein, dass er in der Lage ist eine nützliche, zur weiteren Verarbeitung geeignete Ausgabe trotz fehlerbehafteter Eingabe zu liefern.

Es reicht nicht aus, die Ausgabe eines Parsers mit der Ausgabe eines anderen Parsers zu vergleichen. Da unterschiedliche Parser nicht dieselben Informationen generieren, müsste man für einen solchen Vergleich entweder Informationen hinzufügen oder Informationen entfernen.

Generell kann man unterscheiden zwischen Evaluationsmethoden, die dazu dienen die Entwicklung eines bestimmten Grammatik/Parsing-Systems zu unterstützen und zu überwachen und solchen, die geeignet sind, verschiedene Systeme miteinander zu vergleichen.

Nach Galliers und Sparck Jones [GAL93] gibt es zwei Hauptkriterien bezüglich der Evaluation von Performanz:

> „Intrinsische Kriterien sind die Kriterien, die sich auf die Zielsetzung des Systems beziehen. Extrinsische Kriterien, sind die Kriterien, die sich auf die Funktion des Systems beziehen, zum Beispiel die Rolle des Systems in Bezug auf seine Installationsabsicht."

Generell wird in Fachkreisen akzeptiert, dass eine Evaluation von Parsern durchgeführt werden sollte, indem man die vom Parser generierten Parsingbäume (auch als Antworten bezeichnet) mit manuell konstruierten Parsingbäumen (auch Schlüssel genannt) vergleicht [LIN95]. Eine intrinsische Evaluation würde also die Treffgenauigkeit bzw. die Fehlerfreiheit eines Parsers in einem eigenständigen System untersuchen, wobei eine extrinsische Evaluation den Einfluss des Parsers im Kontext einer NLP-Applikation analysiert. Die meisten Ansätze untersuchen, wie auch diese Arbeit, den Parser unabhängig. Nur in den wenigsten Fällen wurde bisher der Effekt von verschiedenen Parsern in echten Applikationen erfasst.

Bestehende Parser- und Grammatik-Evaluationsmethoden unterteilen sich nach [CAR98] außerdem („in

- korpus-basierte und
- nicht-korpus-basierte

Methoden. Wobei sich die korpus-basierten Methoden zusätzlich in

- nicht-annotierter Korpus-basierte und
- annotierter Korpus-basierte

Methoden unterteilen.") Ein paar der gängigsten Evaluationsmethoden für Parser werden im Folgenden kurz vorgestellt und erläutert.

Parseval

Die Grammar Evaluation Interest Group (GEIG) hat eine Evaluationstechnik vorgeschlagen, die sich *Parseval* nennt [HAR91]. Diese Methode vergleicht Phrasenstrukturklammerungen, die vom Parser produziert wurden, mit Klammerungen aus einer Treebank wie zum Beispiel TIGER [BRA02] und berechnet die Anzahl der übereinstimmenden Klammerungen *M*, in Bezug auf die Anzahl der Klammerungen *P*, die der Parser liefert (bezeichnet als *Precision M/P*) und das Verhältnis zwischen den Klammerungen *M* zu den Klammerungen im Korpus *C* (bezeichnet als *Recall M/C*). C wird auch *Gold Standard* genannt.

Zusätzlich wird die Anzahl der sich überschneidenden Klammern berechnet. Das bedeutet, die Anzahl der sich überschneidenden Klammern zwischen dem, was der Parser liefert und dem Gold Standard.

Ein Vorteil dieser Methode ist, dass man lediglich eine geklammerte Baumbank-Darstellung benötigt. Ein Nachteil von *Parseval* ist jedoch, dass ein einzelner Fehler mehrmals gezählt werden kann. Dazu ein Beispiel:

a) [I [saw [[a man] [with [[a dog] and [a cat]]]] [in [the park]]]]

angenommen dies sei der Schlüssel, und die Ausgabe

b) [I [saw [[a man] [with [[a dog] and [[a cat] [in [the park]]]]]]]]

sowie die Ausgabe

c) [I [saw [a man] with [a dog] and [a cat] [in [the park]]]]

seien Antworten, die von zwei verschiedenen Parsern generiert werden.

1 Fehler: PP-Zuweisungsfehler bei dem Wort cat statt bei dem Wort saw, dennoch gibt es drei sich überschneidende Klammern:

1. [a dog and a cat] vs. [a cat in the park]
2. [with a dog and a cat] vs. [a dog and a cat in the park]
3. [a man with a dog and a cat] vs. [with a dog and a cat in the park]

Recall: 7/10= 70%, Precision: 7/11=63.6%

Im Gegensatz dazu, hat die stellt die Ausgabe

c) [I [saw [a man] with [a dog] and [a cat] [in [the park]]]]

eine sehr flache Analyse des Satzes dar. Es gibt keine sich überschneidenden Klammern.

Recall: 7/10=70%, Precision: 7/7=100%

Obwohl die Struktur b) offensichtlich mehr Gemeinsamkeiten mit a) aufweist als c), wird b) aufgrund der Werte von Precision, Recall und der Anzahl der sich überschneidenden Klammern, schlechter bewertet. Außerdem besteht weiterhin das Problem unterschiedliche Annotationsschemata miteinander zu vergleichen.

Abgedeckte linguistische Konstruktionen (kein Korpus)

Der traditionelle Ansatz einen Parser zu evaluieren, besteht daraus, einfach Konstruktionen, die abgedeckt und „hoffentlich“ auch nicht von einer gegebenen Grammatik oder einem gegebenen Parser abgedeckt wurden (für Beispiele siehe [ALS92] und [BRI87]) aufzulisten.

Der Vorteil dieser Methode liegt darin, dass man keinen Korpus benötigt.

Nachteil dieser Methode ist jedoch, dass das Zusammenspiel linguistischer Konstruktionen nicht berücksichtigt wird. Ein großer Anteil der Komplexität von linguistischen Konstruktionen jedoch, kann jedoch in eben diesem Zusammenspiel der Konstruktionen liegen. Verfolgt man zum Beispiel das Ziel, englische Relativsätze abzudecken, kann es vorkommen, dass man es versäumt hat, Fälle abzudecken, die resumptive Pronomen enthalten:

*a poorly-drafted law which it is difficult to see how **it** could be enforced*

Ähnlich ist es bei der Abdeckung von Koordinationen und dem verbalen Komplement, die ebenfalls keine Abdeckung ihres Zusammenspiels miteinander garantiert.

Abdeckung (nicht-annotierter Korpus)

Die Idee hinter diesem Verfahren ist, dass es ein nützlicher erster Schritt für eine Evaluation ist, den Anteil der Sätze des Korpus zu errechnen, für die der Parser eine oder mehrere Analysen liefert [BLA93] [BRI95]. Dies kann für einen großen Korpus berechnet werden (wenn man einen Parser hat, der effizient genug ist) und man benötigt keine Annotation des Korpus. Allerdings ist dieses Maß ein sehr schwaches Qualitätsmaß, da es keine Garantie dafür gibt, dass die gefundenen Analysen korrekt sind. Das bedeutet, dass es eine unbekannte Anzahl von falschen Positiven im Resultat gibt, die sich nur finden lassen, indem man den Output manuell kontrolliert. Außerdem ist es als einziges Kriterium anfällig dafür, missbräuchlich angewandt zu werden: für die triviale Grammatik S ----> Wort * würde dieses Maß zum Beispiel eine perfekte Auswertung bekommen.

Durchschnittliche Parsing-Basis (Average Parse Base APB) (nicht- annotierter Korpus)

Black, Garside & Leech (1993) definieren die Parsing-Basis (Parse Base, PB) einer Grammatik als das geometrische Mittel der Anzahl der Analysen, geteilt durch die Anzahl der Input-Token in jedem geparsten Satz. Dies ergibt ein Maß, aus dem man die durchschnittliche Anzahl der (nicht-ambiguierten) Parses für einen Satz, der eine Länge von n Wörtern hat, aus einem bestimmten Korpus stammt und mit dieser Grammatik/diesem Parser analysiert wurde, voraussagen kann. Das Problem bei dieser Annahme ist, dass durch die Mittelung der Parses im Verhältnis zu den Tokens im Satz, implizit angenommen wird, dass die Anzahl der Analysen fast linear mit der Länge der Sätze anwächst. In der Praxis ist das Verhältnis aber eher exponentiell , was dazu führt, dass das PB-Maß die erwartete Anzahl der Parses für kürzere Sätze überschätzt, und die Anzahl der Parses für längere Sätze unterschätzt. Deshalb schlagen Briscoe & Carroll (1995) eine überarbeitetes Maß, die durchschnittliche Parsing Basis vor: „die geometrische Mitte aller Sätze im Korpus von $\sqrt[n]{p}$, wobei n die Anzahl der Wörter eines Satzes ist, und p, die Anzahl der Parses für diesen Satz. Folglich liefert die durchschnittliche Parsing Basis hoch n die Anzahl der Parses, die für einen Satz der Länge n erwartet werden würde. Für dieses Maß gilt, für einen Parser der effizient genug ist, kann die APB, sogar für große Korpora einfach berechnet werden und liefert ein knappes Maß des Grades an Ambiguität innerhalb einer Grammatik. Die Effizienz bezeichnet das Kosten-Nutzen-Verhältnis bei der Anwendung des Verfahrens [FUH06]. Ein großer Nachteil liegt darin, dass es nicht möglich ist, die Leistungsfähigkeit verschiedener Parser auf verschiedenen Daten sinnvoll zu vergleichen, da die Ambiguität der Daten und die der Grammatik sich gegenseitig beeinflussen.

Entropie (nicht-annotierter Korpus)

Ein propabilistisches Sprachmodell kann dazu genutzt werden, die Entropie eines Korpus (oder äquivalent dazu der Informationsgehalt des Modells) zu berechnen [CAR98]. Das führt zu der Gradzahl, zu der ein Modell Regelmäßigkeiten im Korpus durch Minimierung von Unvorhersagbarkeit und Ambiguität erfasst. Obwohl dieses Maß in seiner ursprünglichen Form auf ein gegebenes propabilistisches Sprachmodell und Korpus bezogen ist, kann diese Annäherung dazu genutzt werden, die Effektivität verschiedener Sprachmodelle oder die Effektivität verschiedener Trainingssysteme zu vergleichen. Außerdem kann dieser Ansatz dazu genutzt werden, durch sukzessive Approximation mittels N-Gramm-Sprachmodellen[2] für immer größere Werte von n, ein Modell-unabhängiges Maß für die Komplexität eines Korpus (Sharman, 1990) zu bestimmen. Diese Methode, eine verallgemeinerte Entropie zu berechnen, ist sehr „kostenintensiv“ in der Berechnung. „Haben alle Sätze geringe Wahrscheinlichkeiten, ist der Parser eher schlecht. Haben alle Sätze eine hohe Wahrscheinlichkeit, ist der Parser <1,S> > Wort* oder zumindest eher schlecht [TEI07].“Ein Vorteil ist, dass das Maß innerhalb des propabilistischen Kontextes klar definiert ist. Dadurch ist es möglich eine aussagekräftigen Vergleich zwischen verschiedenen propabilistischen Modellen innerhalb eines Korpus anzustellen.

Es ist allerdings negativ zu bewerten, dass dieses Maß nur in Zusammenhang mit propabilistischen Modellen anwendbar ist. Schließlich muss hinzugefügt werden, dass für eine gültige und sinnvolle Evaluation eines Parsers unbedingt auch die Arbeit des vorangegangenen Part of Speech Taggers untersucht und bewertet werden muss. Die Arbeit des Parsers hängt grundlegend von der des genutzten Taggers ab.

2.5 Zusammenfassung

Die beiden nachfolgenden Tabellen fassen die verschiedenen vorgestellten Methoden und Verfahren und die damit verbundenen Möglichkeiten abschließend noch einmal, getrennt zwischen POS-Tagging und Parsingmethoden zusammen. Anschließend wird daraufhin die Motivation für die Wahl von SMES als geeignetes System aus einer grossen Vielzahl von linguistischen Möglichkeiten begründet.

2 N-Gramm - ordnet jedem Wort eine Wahrscheinlichkeit $P(w_i \mid w_{i-(n-1)}...w_{i-1})$ zu und berücksichtigt dabei die n vorhergehenden Wörter

Part Of Speech Tagger	Merkmale und Arbeitsweise	Vor- und Nachteile
Regelbasierte Tagger	- basieren auf Regeln, die Beschränkungen ausdrücken - ein Wort kann mehrere Bedeutungen im Lexikon haben - Eingabe besteht aus ungetaggtem Text und einem Tagset - alle möglichen Tags werden eingetragen - Regeln zur Disambiguierung	- Lexikon ist nie ganz vollständig - Interaktion der Regeln schlecht überschaubar - für jede Sprache muss ein neues Regelsystem erstellt werden - enormer materieller und intellektueller Aufwand zur Erstellung der Regeln notwendig
Stochastische Tagger	- basieren auf Wahrscheinlichkeiten von Token/Tag-Abfolgen - für mehrdeutige Token wird das jeweils wahrscheinlichste ausgewählt - Nutzung von bedingten Wahrscheinlichkeiten/Markov Annahme - Erstellung einer Wissensbasis - Nutzung von Korpora die von Hand annotiert und mit korrektem POS versehen wurden	- kein Aufstellen komplexer Regelsysteme um Mehrdeutigkeiten zu eliminieren - Wahrscheinlichkeit ist nicht immer Sicherheit - stark abhängig vom Trainingskorpus
Brill Tagger	- basiert auf TBL - Kombination von regelbasierten und stochastischen Verfahren und getaggtem Korpus - betrachtet zur Entscheidung der wahrscheinlichsten Wortklasse nur aktuelles Wort und näheren Kontext - lernt Regeln und statistische Werte aus manuell korrigiertem Trainingskorpus selbstständig	- weniger zu speichernde Information als bei stochastischem Ansatz - weniger als 100 Regeln notwendige - benötigt kein externes Lexikon - Verlangsamung aufgrund unnötiger Operationen in der kontextuellen Phase

Parser	Merkmale und Arbeitsweise	Vor- und Nachteile
Vollständiges Parsing	- vollständige und korrekte Analyse (grammatische Abbildung) als Ziel - „ganz oder gar nicht“-Eigenschaft	- höhere Korrektheit - komplexere Syntaxregeln - sinnvoll bei begrenzter Anwendungsdomäne und wohlgeformten Sätzen - durch Anspruch auf Vollständigkeit nimmt die Effizienz ab
Partielles Parsing	- Ziel ist eine breite Datenabdeckung - nur bestimmte Teilstrukturen eines Satzes werden erkannt	- einfacher Konstituentenaufbau, dadurch schnell und zuverlässig identifizierbar - robust aber weniger korrekt - weniger Syntaxregeln - sinnvoll bei großen, nicht geformten Satzstrukturen
Chunk Parsing	- spezielle Art partielle zu parsen - unvollständiges, lokaes, nicht-rekursives Verfahren - Definition von Chunks, Informationssegmenten - Definition rein auf syntaktischer Ebene - Unterteilung des Analyseprozess in 2 Stufen (Chunker, Attacher)	- Anzahl der Lösungen geringer als bei vollständigem Parsing - kann mit regulären Ausdrücken realisiert werden - keine syntaktischen Attachement-Mehrdeutigkeiten - aufwendige Techniken nur beim Attacher - man kann den Chunker unabhängig vom Attacher entwickeln

3 Syntaktisches Natural Language Processing mit SMES

Im Rahmen dieser Diplomarbeit wurde das ***Saarbrücker Message Extraction Systems (SMES)*** [Neumann et al., 1997, Neumann und Mazzini, 1999] genutzt, um die Texte der Schulkinder zu verarbeiten. Im Folgenden wird dieses System und seine Arbeitsweise genauer erklärt.

Am „Deutschen Forschungsinstitut für Künstliche Intelligenz“ (DFKI) in Saarbrücken gibt es verschiedene Information-Extraction- Projekte. Eines davon ist PARADIME, Parameterizable Domain-Adaptive Information and Message Extraction. [Neumann, Declerk, Piskorski, Uszkoreit, 1999]. PARADIME stellt eine Reihe von robusten und effizienten Natural Language-Komponenten und Werkzeugen zur Erstellung linguistischer Wissensbasen bereit, mit deren Hilfe konkrete Information Extraction- Applikationen entwickelt werden können.

Das Saarbrücker Message Extraction System (SMES) ist in dem Projekt *PARADIME* implementiert und frei erhältlich [NEU06]. SMES kombiniert ein Kernsystem für flaches Parsen und ein benutzerdefiniertes System endlicher Automaten, welches die Grammatik definiert. Die Hauptuntersuchungsgegenstände, mit denen sich [Neumann et al., 1997] befasst haben, beinhalten einfache Portabilität und Adaptivität des Kernsystems. Das Basis-Designkriterium des SMES-Systems ist das Bereitstellen einer Menge grundlegender, robuster und effizienter „Natural Language Komponenten“ und allgemeiner linguistischer Wissensbasen, die einfach angepasst werden können, um verschiedene Aufgaben flexibel verarbeiten zu können. Die Anpassung wird folgendermaßen erzielt:

- Definition der Flusskontrolle zwischen den Modulen (zum Beispiel stufenförmig und/oder überlappend)
- Auswahl der linguistischen Wissensursprünge
- Spezifizierung von domänenspezifischem Wissen
- Definition von zusätzlichen aufgabenspezifischen Funktionalitäten

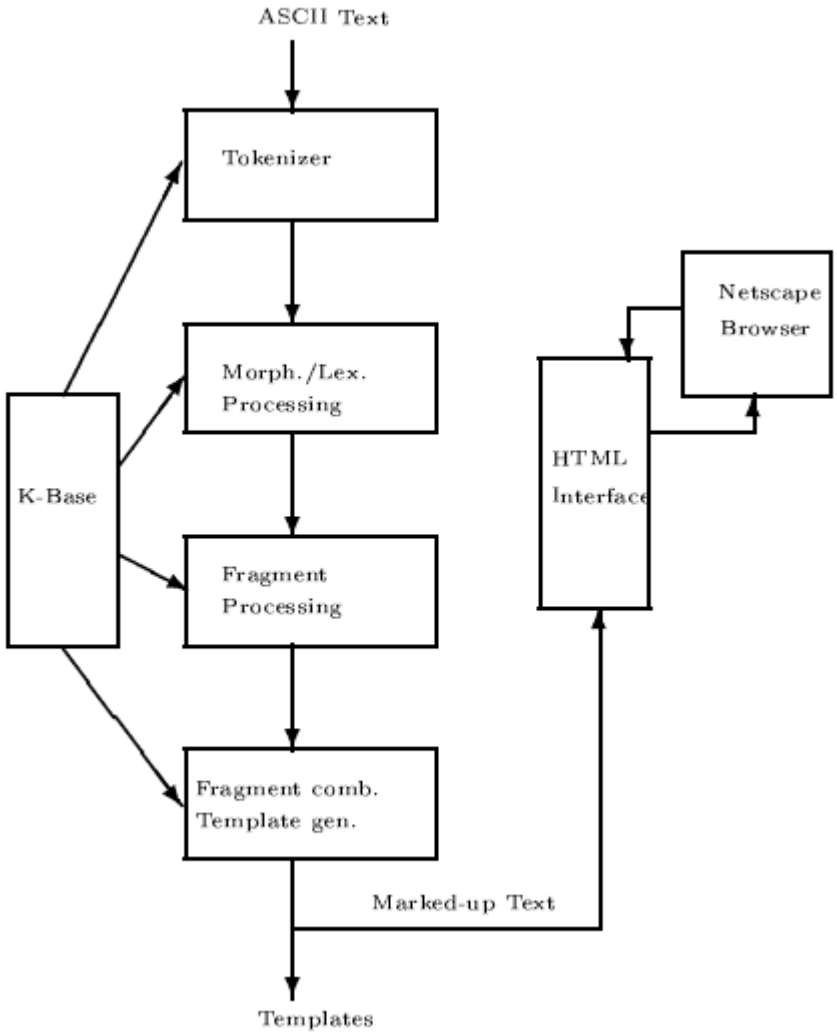


Abb. 3.1 Entwurf eines Kernsystems

Abbildung 3.1 zeigt einen Entwurf des Kernsystems. Die Hauptkomponenten sind:

- Ein Tokenizer, der auf regulären Ausdrücken basiert. Er scannt einen ASCII Text, um die Textstruktur zu erkennen, spezielle Zeichen wie Angabe des Datums, Abkürzungen und Wörter.
- eine sehr effiziente und robuste morphologische Komponente. Für jedes analysierte Wort gibt es (eine Menge) Tripel zurück, die den Stamm, die Wortart und die Flexionsinformation enthält
- die Disambiguierung der morphologischen Ausgabe wird von einer Menge Regeln und einem Brill-basierten Tagger übernommen

- ein deklaratives Spezifikationstool, um reguläre Grammatiken auszudrücken, um Wortgruppen und phrasale Einheiten zu bearbeiten (zum Beispiel NPs, PPs, Verbgruppen)
- ein bidirektionaler lexikon-gesteuerter flacher Parser für die Kombination der extrahierten Fragmente.

Das flache Parsing findet unter der Regie von ***fragment combination patterns*** (**FCP)** der Form <FSTlinks, Anker, FSTrechts>, wobei der Anker ein Lexikoneintrag oder der Name einer Klasse eines Lexikoneintrags ist. FCPs werden an die Lexikoneinträge angefügt und ausgewählt, nachdem ein entsprechender Lexikoneintrag ermittelt wurde. Sie werden an die rechten und linken Zeichenketten der erkannten Fragmente angefügt. FST steht für finite state transducer. FST haben die Form

<identifier, recognition part, output description, compiler options>. Recognition part ist ein regulärer Ausdruck. Die output description ist typenabhängig.

Der Fragmentkombinierer wird sowohl für die Erkennung und Extrahierung von Ausdrücken auf Satzlevel genutzt als auch für die Instantiierung von Dokumentvorlagen. Außerdem gibt es ein Interface für TDL, eine type description language, für constraint-based Grammatiken [Krieger und Schäfer, 1994]. TDL wird in SMES genutzt um eine typ-gesteuerte lexikalische Suche durchzuführen, zum Beispiel für die Evaluation von syntaktischen Übereinstimmungstests während der Fragmentverarbeitung und -Kombination.

Die Wissensbasis wird gebildet durch die Sammlung verschiedener Wissensquellen wie das Lexikon, Subgrammatiken und Vorlagenmuster. Momentan sind 120.000 Lexikongrundeinträge, Subgrammatiken für einfache und komplexe Zeit- und Datumsangaben, Namen von Personen, Firmennamen, Währungsbezeichnungen sowie flache Grammatiken für allgemeine Nominalphrasen, Präpositionalphrasen und allgemeine Verb-Modifier Ausdrücke. Jede Komponente des Systems gibt die aus sich resultierende Struktur uniform zusammen mit der Anfangs- und Endposition des untersuchten Ausdrucks. Jede Struktur aus jeder Komponente wird gespeichert, so dass jede Komponente die Ergebnisse der vorangegangenen Komponenten mit einbeziehen kann.

Die folgende Abbildung zeigt das Zusammenwirken der einzelnen Komponenten und die Arbeitsweise von SMES.

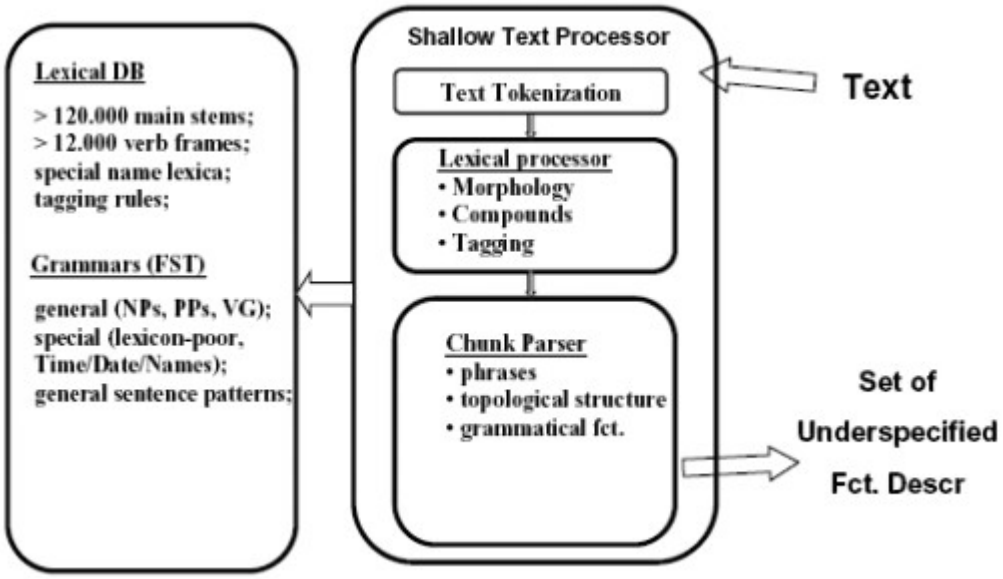


Abb. 3.2 Arbeitsweise und Zusammenwirken der SMES-Komponenten

Im Folgenden werden einige der Architekturkomponenten und die damit verbundenen Arbeitsschritte genauer erklärt.

Vorverarbeitung

Jede Datei wird als erstes von dem Text Scanner vorverarbeitet. Der Text Scanner identifiziert Wörter, Zahlen, Zeit- und Datumsangaben und dehnt Abkürzungen aus. Er gibt eine Zeichenkette aus. Zahlen, Zeit- und Datumsangaben werden normalisiert. Sie sind als Attribut-Wert-Strukturen repräsentiert. An den Text Scanner schliesst die morphologische Verarbeitung an. Die morphologische Komponente MORPHIX, eine schnelle klassifikationsbasierten morphologischen Komponente für das Deutsche [Finkler und Neumann, 1988]. MORPHIX gibt eine Liste von Token aus, die aus einer Wortform mit ihren verschiedenen Lesearten besteht. . Eine Lesart wird charakterisiert durch ein Tripel der Form <stem, inflection,pos>, wobei stem ein String oder eine Liste mit Strings ist, inflection ist die Flexionsinformation und pos steht für eine Kategorie. Im letzten Schritt wird diese Liste von MORPHIX-Token anhand der Satzzeichen in einzelne Sätze aufgeteilt. Diese können dann iterativ an den Satzparser weitergegeben werden.

Morphologisch mehrdeutige Versionen werden disambiguiert, durch den Gebrauch von Filter-Regeln, die mittels des Brill-Taggers ermittelt wurden und den Gebrauch von Fall-abhängigen Regeln. Zum Beispiel kann eine Filterregel folgendes besagen: Ändere den Tag einer Wortform in Nomen, wenn der Vorgänger ein Artikel ist.
Nach dem Scannen und der morphologischen Analyse der identifizierten Fragmente findet flaches Parsen in zwei Schritten statt. Spezifizierte finite state transductors (FST) führen auf der Basis des Morphix-Outputs eine Fragmenterkennung und -Extrahierung durch. Die zu identifizierenden Fragmente sind benutzerdefiniert und bestehen typischerweise aus phrasalen Einheiten (zum Beispiel NPs, PPs) und anwendungsspezifischen Einheiten (zum Beispiel Uhrzeit, Datum).

In einem zweiten Schritt operieren benutzerdefinierte Automaten, die Verb-Fragmente repräsentieren, auf den bereits extrahierten Fragmente, um Komplemente und Adjunkte zu Prädikat-Argument Strukturen zu kombinieren.

4 Konzeption der Evaluationsstudie und ihre Umsetzung

Im folgenden Kapitel wird genauer auf den Aufbau und die Vorgehensweise der durchgeführten Evaluation eingegangen. Dazu wird zunächst die Wahl des ausgewählten Parsers begründet. Anschließend wird das Evaluationskorpus detailliert beschrieben und die notwendigen Vorverarbeitungsschritte für das Durchführen einer Evaluation erläutert.

Schließlich werden die Ergebnisse, die SMES liefert dargestellt. Entstandene Fehler werden genauer beleuchtet, mögliche Gründe dafür erwähnt. Abschließend wird eine, aufgrund der gewonnenen Erkenntnisse, getroffene Auswahl an nützlich und sinnvoll erscheinenden Modifikationen, und die Ergebnisse die damit erzielt wurden vorgestellt und diskutiert.

4.1 Entscheidungskriterien für die Parserwahl

Im Vorfeld der Evaluation der Arbeitsweise von SMES in Bezug auf die Kindergeschichten fand ein ausführliches Auswahlverfahren geeigneter linguistischer Methoden, bzw. speziell für den gegebenen Kontext geeignete, bereits existierende Parsingstrategien statt.

Die Wahl des Parsers hing von mehreren Faktoren ab, auf die im folgenden näher eingegangen wird. Als Eingabe dienen dem die Tool transkribierten Texte. Es wurden zwar einige wenige Korrekturen im Vorfeld der Verarbeitung der Texte vorgenommen, wie im nächsten Kapitel genauer erläutert wird, jedoch war die Bestrebung, das Ausmaß solcher Korrekturen so gering wie möglich zu halten. Die Fähigkeit, mit der schwierigen und fehlerbehafteten Art der Eingabe arbeiten zu können, war also ein sehr wichtiges Kriterium für die Wahl des Parsers.

Andere Kriterien, die in diesem Zusammenhang wichtig sind, sind die Verlässlichkeit eines Tools und Anhand der Ausgabe, die SMES liefert, lassen sich interessante, und für das langfristige Ziel, die Entwicklung eines Programms, für den aktiven Einsatz im Unterricht, nützliche Informationen ablesen. Es ist zum Beispiel möglich, eine Aussage über das jeweilige Entwicklungsniveau in Bereichen wie der Verbstellung innerhalb des Satzes, Interpunktion und schließlich allgemeine Rechtschreibung der SchülerInnen zu machen. SMES bietet durch die Möglichkeit, verschiedene Arten von Ausgaben zu generieren, dem Nutzer/der Nutzerin die Option, sich eine Ausgabe generieren zu lassen, die das individuelle Anliegen bzw. Interesse so direkt wie eben möglich befriedigen. Es ist möglich, den Fokus auf einen bestimmten Aspekt, wie zum Beispiel die „grammatikalische Funktion“ zu legen.

Erwähnt werden sollte in diesem Zusammenhang auch der Aspekt der Benutzerfreundlichkeit. Die Benutzerfreundlichkeit lässt sich in diesem Kontext auf zwei Arten bewerten. Außerdem ist es wünschenswert, dass das System sich für einfache Bedürfnisse, preis- und kostengünstig erweitern bzw. modifizieren lässt. Einen Anspruch, den SMES, bis zu einem gewissen Grad, zu befriedigen in der Lage ist. Bei SMES besteht die Möglichkeit, das Vokabular, auf individuelle Bedürfnisse hin, einfach und schnell erweitern zu können. Dadurch ist es möglich, domänenspezifisches Vokabular und Eigennamen problemlos zu integrieren. Ein Vorteil, der speziell im Schulunterricht sehr nützlich ist. Dadurch bietet sich die Möglichkeit, Begriffe und eventuell zu erwartende Rechtschreibfehler in diesem Zusammenhang zu tolerieren, wenn dies gewünscht sein sollte.

Zunächst einmal galt es, die besondere Textsorte, die Aufsätze der Kinder, genauer zu betrachten. Durch die Vielzahl an Fehlern, die die Kinder in ihren Niederschriften aufweisen, sowohl die Variation, als auch die Häufigkeit betreffend, ist die Entscheidung gefallen, eine partielle, flache Analyse durchzuführen.

4.2 Durchführung der Studie und ihre Voraussetzungen

Das Projekt VERA entstand 2005 und wurde durchgeführt von dem Fachbereich Psychologie der Universität Koblenz-Landau, Campus Landau. Das Projekt umfasst Fragen auf mehreren schulischen Gebieten. Zunächst wurden den Schülern Fragen nach dem Alter, Geschlecht und mehr gestellt. In diesem Kontext interessieren nur die Fragen, die spezifisch im Block der Deutschaufgaben gestellt wurden.

Den Kinder wurden mehrere Aufgaben zur Bearbeitung gegeben, die innerhalb von zwei Schulstunden zu lösen waren. Da die Aufgaben diesen zeitlichen Rahmen bewusst überschritten, war die Bitte an die Kinder, so viele Aufgaben wie möglich zu bearbeiten. Die Aufgaben umfassten verschiedene Gebiete des Schulstoffs. Darunter sowohl Gebiete, die die Kinder bereits im Unterricht bearbeitet hatten, als auch Gebiete, die noch nicht im Unterricht behandelt wurden.

Den Kindern wurden zur Bearbeitung der gestellten Aufgaben Hilfestellungen gegeben. Zum Beispiel, wie die Kinder vorgehen sollen, wenn sie sich verschrieben haben. Die Aufgabe, die im Rahmen dieser Diplomarbeit untersucht wurde wird von nun an im weiteren Kontext „Raketengeschichte“ genannt. Sie ist die erste Aufgabe der Aufgabensammlung Deutschaufgaben gewesen.

Die Aufgabenstellung bestand darin, aus einem gezeigten Bild mit einer Überschrift und gegebenen Schreibhilfen eine Geschichte zu konstruieren und aufzuschreiben. Die Abbildung 4.2.1 zeigt das Bild mit der vorgegebenen Überschrift.

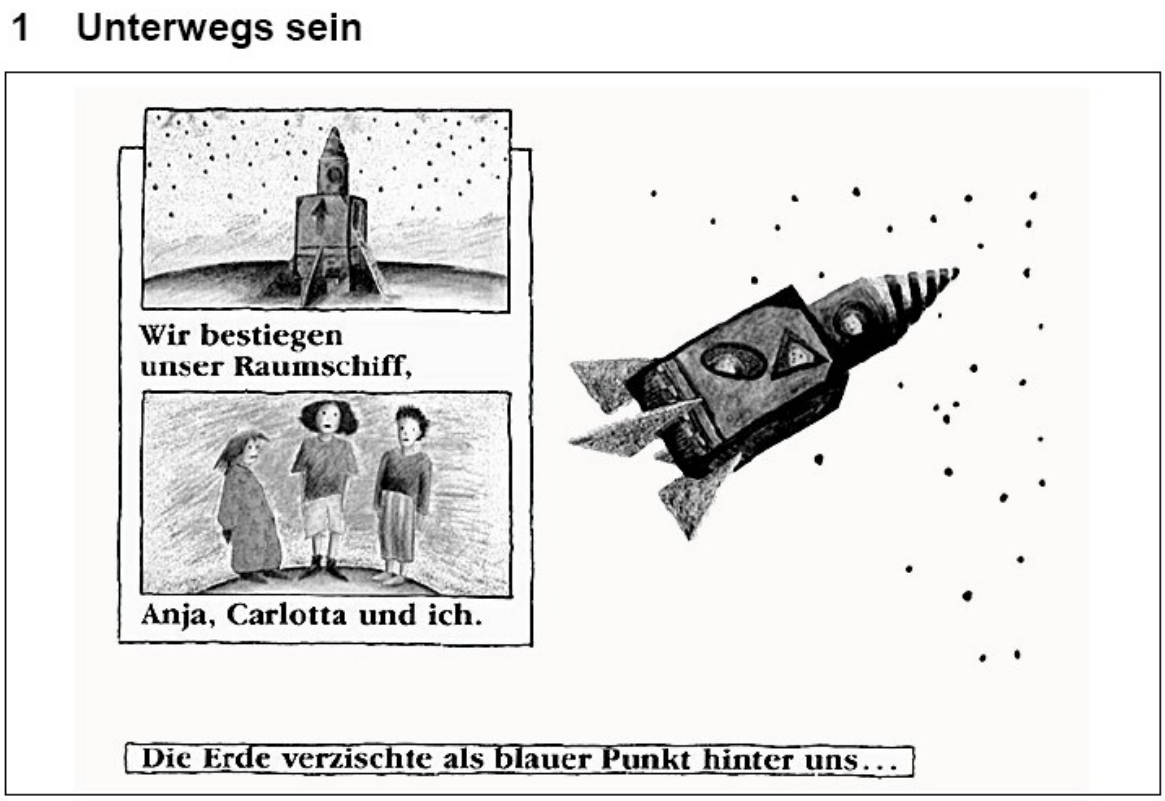


Abb. 4.2.1: Bild zur Geschichte

Die genaue Aufgabenstellung für die Kinder zu diesem Bild sah folgendermaßen aus:

Abb. 4.2.2: Angebotene Schreibhilfen

Auf der nächsten Seite wurden den Kinder mehrere Linien aufgezeichnet, auf denen sie ihre Geschichte niederschreiben konnten. Zusätzlich zu der Aufgabenstellung, wurden den Kindern noch Schreibhilfen zur Verfügung gestellt:

Die Aufsätze, die die Kinder geschrieben haben wurden anschließend auf vielfältige Art und Weise untersucht. Außerdem wurden die von den Kindern geschriebenen Geschichten verschiedenen Korrekturformen unterzogen.

Zunächst lagen die Daten in tabellarischer Form vor. Die Tabelle besteht aus 4 Spalten. Die erste Spalte wird mit ID1 bezeichnet und nummeriert die folgenden Geschichten von 1 aufwärts durch. Diese ID wurde übernommen und bezeichnet im Verlauf dieser Arbeit jeweils die ID der Raketengeschichte, die in diesem Kontext untersucht wurde. Die zweite Spalte trägt die Bezeichnung ID und steht für die Identifikationsnummer, die die Kinder während des Ablaufs der Arbeit automatisch zugewiesen bekommen haben. Jedes Kind hat anhand seines Aufgabenblattes eine eindeutige ID bekommen, die jedoch aus Gründen der Anonymität keine Rückschlüsse auf die konkreten Daten des Schülers/der Schülerin, wie zum Beispiel den Namen, schliessen lässt. Ein Zweck dieser Nummer bestand darin, anschließend einfacher und schneller feststellen zu könne, welches Kind die Aufgaben überhaupt bearbeitet hat. Die Überschrift der dritten Spalte setzt sich aus einer Kombination von Buchstaben und Zahlen zusammen, die in diesem Zusammenhang nicht näher erläutert wird. Inhalt dieser Spalte ist die jeweilige Überschrift, die die Kinder ihren Aufsätzen gegeben haben.

Das bedeutet, dass diese Spalte in einigen Fällen leer ist, da nicht jedes Kind eine Überschrift definiert hat.

Die letzte Spalte gestaltet sich in ihrer Überschrift ähnlich der vorhergehenden und beinhaltet die von den Kindern geschriebene Geschichte zu dem vorgegebene Thema, das heißt zu den vorgegebene Bildern und den bereits erwähnten Hilfestellungen.

Von dieser Art Tabelle gab es zunächst zwei verschiedene Fassungen. In der ersten Fassung standen die Geschichten in der vierten Spalte tatsächlich so, wie die Kinder sie zu Papier gebracht hatten. Das bedeutet ohne jede Form von Korrektur oder Veränderung. Anschließend wurden verschiedene Korrekturmaßnahmen vorgenommen, um eine einigermaßen brauchbare Basis zu erhalten. Es ist jedoch von Bedeutung festzuhalten, dass auch die teilweise korrigierten Texte nicht fehlerfrei sind. Diese Ergebnisse wurden separat in einer zweiten Tabelle „Rakete korrigiert“ festgehalten.

Um den Unterschied der Texte in Rohfassung und nach durchgeführten Korrekturen deutlich zu machen dient folgendes Beispiel. Dazu der Inhalt der vierten Spalte zu ID 4, welche im folgenden auch als Raketengeschichte 4 (abgekürzt rak4) bezeichnet wird zunächst ohne Korrektur:

Heute fliegen ins welt all. Die Kind freuen sih sehn drauf; Jetzt ist es soweit. Die Kind gehen in die Rakete rein und 109876543210 und Start die Kind sint gestatet, Jetzt sint sie am Mond und fliegen wird zurüg.

Nun mit den vorgenommenen und teilweise gekennzeichneten Korrekturen:

Heute fliegen ins Weltall. Die Kind freuen sich sehr drauf; Jetzt ist es soweit. Die Kind gehen in die Rakete rein und 109876543210 und Start die §[Kind]§ Kinder sind gestartet, Jetzt sind sie am Mond und fliegen §[wird]§ wieder zurück.

Daraus lässt sich sowohl die ursprüngliche Fassung des Textes, als auch die bereits in einem ersten Verarbeitungsschritt vorgenommenen Korrekturen ablesen:

In den eckigen Klammer eingerahmt von dem Paragraphen steht das ursprünglich geschriebene , aber in diesem Kontext falsche Wort, anschließend das Wort, durch welches der Fehler ersetzt wurde.

Für die vorgenommenen Korrekturen ist die Arbeitsgemeinschaft aus Landau verantwortlich, jedoch wurde sich in gemeinsamen Treffen, zwischen den Arbeitsgruppen aus Koblenz und Landau, die Auswahl über eine sinnvolle Korrektur und die sinnvolle Kennzeichnung einer solchen Korrektur besprochen und geeinigt.

Für die anschließende Verarbeitung der Texte im Rahmen dieser Diplomarbeit wurden die Texte noch einmal separat gespeichert.

Dazu wurde unter Bezeichnung der ID, die Überschrift der Geschichte, falls vorhanden und der jeweilige Inhalt der Geschichte (bis jetzt Inhalt der vierten Spalte), mit direkter Übernahme der Korrekturen jedoch ohne Kennzeichnung der Korrektur übernommen. Gespeichert wurde der Text in einem für die Verarbeitung durch Parser gängigen Format, das bedeutet pro Zeile ein Satz. Orientiert wurde sich dabei an der Setzung der Satzzeichen, wie sie durch die Kinder vorgenommen wurde. Das bedeutet zum Beispiel für die Raketengeschichte 4, dass sie in folgendem Format gespeichert und an dem Parser so als Eingabe übergeben wurde:

Heute fliegen ins Weltall .
Die Kind freuen sich sehr drauf ; Jetzt ist es soweit .
Die Kind gehen in die Rakete rein und 109876543210 und Start die Kinder sind gestartet , Jetzt sind sie am Mond und fliegen wieder zurück .

4.3 Evaluation der Arbeitsweise von SMES in Bezug auf die transliterierten Schulaufsätze

In diesem Kapitel werden die verwendeten Schemata und Verfahren vorgestellt. Zunächst wird die Leistungsfähigkeit des Parsers im gegebenen Kontext anhand von Qualitätsmerkmalen klassifiziert, danach werden die aufgetretenen Fehler und Mängel nach Art und Ursache detailliert analysiert. Anschließend werden sinnvoll erscheinende Verbesserungsvorschläge und Modifikationen vorgestellt.

Die wohl am häufigsten verwendeten Maße zur Beurteilung der Güte eines Information Retrieval Systems sind Recall und Precision. Precision und Recall haben einen feststehenden Wertebereich und können leicht miteinander verglichen werden.

Der Recall stellt die Vollständigkeit eines Rechercheergebnisses dar. Er gibt an, wie gut die Fähigkeit des Systems ist, die richtigen Antworten zu finden und ist definiert als das Verhältnis zwischen den gefundenen relevanten Dokumenten und der Gesamtanzahl der in der vorliegenden Dokumentenbasis vorhandenen relevanten Dokumente.

Die Precision dient zum Messen der Genauigkeit der Recherche. Sie beschreibt die Fähigkeit des Systems, unerwünschte Ballastdokumente, also nicht-relevante Dokumente auszuschliessen. Sie ist definiert als das Verhältnis der gefundenen relevanten Dokumente zur Zahl aller gefundenen Dokumente.

Beide Maße können Werte zwischen Null und Eins annehmen (das entspricht 0% bis 100%) und hängen voneinander ab. Ein Recall von Null wird für das schlechteste Ergebnis, Eins für das beste Ergebnis vergeben. Auch bei der Precision wird versucht den Wert zu maximieren. Die Ballastquote (noise ratio) ist die Anzahl der nicht relevanten nachgewiesenen Dokumente, verglichen mit allen nachgewiesenen Dokumenten (b/S) [EIB02].

Da der Recall die Ballastquote nicht berücksichtigt, kann er leicht auf Eins gesetzt werden, indem alle Dokumente, die sich im Dokumentbestand befinden, nachgewiesen werden. Dann wäre der Precision-Wert allerdings sehr niedrig. Die Precision könnte dadurch maximiert werden, dass sehr wenige Dokumente nachgewiesen werden. Würde man aber nur die Precision betrachten, bekäme man keine Aussage über die Vollständigkeit des Ergebnisses.

Messungen von Recall und Precision können immer dann durchgeführt werden, wenn das Gesamtergebnis eines Systems sich in klar definierbare, voneinander abgegrenzte Teile (Antworten) zerlegen lässt, wie zum Beispiel die einzelnen Wort-Tag-Paare bei Taggern oder die einzelnen Konstituenten bei Parsern.

Die Vorteile der Standardmaße Recall und Precision sind nach [WOM89]

- die weite Verbreitung und
- die einfache Interpretierbarkeit.

Aus diesen Gründen werden die beiden Maße auch als Beurteilung der Ergebnisse dieser Arbeit herangezogen. Die Evaluation wurde geschichtenweise und satzweise vorgenommen.

4.3.1 Evaluation der Arbeitsweise von SMES ohne vorherige Veränderungen

Für die vorliegende Evaluation wurden insgesamt 76 Geschichten zu einem vorgegebenen Bild zusammengestellt. Die Geschichte, die zur Evaluation genutzt wurde, trägt den Titel Raketengeschichte. Wie in Kapitel 2.4 bereits beschrieben, wurden die Daten auf verschiedene Art und Weise vorverarbeitet, um sie für die Verarbeitung mit SMES brauchbar zu machen.

Zusammenfassend können folgende statistischen Angaben gemacht werden:

- die durchschnittliche Länge der Geschichten beträgt 7,6 Sätze,
- die längste Geschichte enthielt 25 Sätze,
- die kürzeste Geschichte bestand aus nur einem Satz,
- insgesamt umfassten alle Geschichten zusammen 578 Sätze,
- 5 der Geschichten wurden von SMES gar nicht geparst. Das bedeutet, dass SMES für 94,74% der Geschichten eine Ausgabe geliefert hat. (Das bedeutet, dass zumindest „etwas“ verarbeitet wurde).
- Von den insgesamt 578 Sätzen wurden insgesamt 299 Sätze von SMES verarbeitet. Das bedeutet, dass SMES für 51,73% aller Sätze eine Ausgabe produziert hat.
- Die größte Parse-Failed-Rate innerhalb der Geschichten lag bei 100%, was nicht überraschend ist, da es bereits wie erwähnt 5 Geschichten gab, die gar nicht von SMES verarbeitet werden konnten.
- Die insgesamt niedrigste Parse-Failes-Rate lag innerhalb der untersuchten Geschichten bei 12,5% (Raketengeschichte 31, 8 Sätze insgesamt, 1 Parse-Failed).

Um die Arbeit von SMES genauer beurteilen zu können, um schließlich heraus zu finden, wo die Stärken und die Schwächen im Umgang mit den Schulkindertexten liegen wurde anschließend der Blick auf die detaillierte Ausgabe gerichtet. Das bedeutet, dass es sich hier teilweise auch um eine diagnostische Evaluation handelt. „Eine diagnostische Evaluation dient dem systematischen Aufspüren und Lokalisierung von Fehlern eines Systems“ [ASC05].

Hier wird noch einmal bewusst, dass es unumgänglich ist, einen beachtlichen Teil an „Händischer“ Analyse zu betreiben, um eine fundierte Aussage über die Arbeit des Systems in Bezug auf die transliterierten Texte machen zu können.

Zunächst wurden dazu alle, in den vorliegenden Geschichten vorhandenen NPs gezählt.

Insgesamt sind in den Texten 273 Nominalphrasen vorhanden. SMES hat insgesamt 163 Nominalphrasen erkannt. Vier Ketten von Wörtern wurden fälschlich als Nominalphrasen analysiert. Das bedeutet, dass insgesamt 159 Nominalphrasen korrekt analysiert wurden. Das entspricht einer Quote von insgesamt 97,55% (= *Precision*). Nicht vom System erkannt wurden 110 Nominalphrasen. Das wiederum entspricht einer Quote von 40,29% an Nominalphrasen, die vom System nicht erkannt wurden. Dadurch lassen sich folgende Aussagen treffen:

Man unterscheidet dazu zunächst zwischen der phrasalen Ebene und der internen Struktur.

In der phrasalen Ebene bezeichnet in diesem Fall der Begriff Nominalphrase die entscheidende Beschreibungseinheit. Man unterscheidet in lexikalische Kategorien, in diesem Fall Nomen, und die dazugehörige phrasale Kategorien, die Nominalphrase [FAN93].

Wendet man nun die bereits erwähnten Erfolgsmaße Recall und Precision an, definieren sich die beiden Werte in diesem Kontext so:

<u>Phrasale Ebene Nominalphrasen:</u>

Recall = Anzahl der richtig identifizierten Nominalphrasen/ Anzahl aller genutzten Nominalphrasen

Precision = Anzahl der korrekt identifizierten Nominalphrasen/ Anzahl aller vom Parser erkannten Nominalphrasen.

Mit den konkreten Werten bedeutet dies:

Recall = 159/273 das bedeutet ca. 0,58

Precision = 159/163 das bedeutet ca. 0,98

Das Gleiche lässt sich für die Präpositionalphrasen durchführen:

Insgesamt lagen 87 Präpositionalphrasen in allen Geschichten vor. SMES hat davon 52 erkannt, 35 hingegen blieben unerkannt vom System. Das bedeutet:

Phrasale Ebene Präpositionalphrasen:

Recall = 52/87 das bedeutet ca. 0,6

Precision = 52/52 entspricht einem Wert von 1

Für eine Aussage über die interne Struktur wird die ***Head-Modifier-Struktur*** untersucht.

Head bezeichnet das Nomen eines zweigliedrigen Kompositums, das dessen wesentliche Bedeutung ausdrückt (z.B. Haustür, Türschloss).

Der *Modifier* spezialisiert oder modifiziert die Bedeutung des Heads. In dem Ausdruck

a very fast car bezeichnet

car den head und

a very fast bezeichnet den modifier.

Interne Struktur:

Recall = Anzahl der Nominalphrasen mit richtiger Zuweisung/ Anzahl der korrekt identifizierten Nominalphrasen

Precision = Anzahl der Nominalphrasen mit richtiger Zuweisung/ Anzahl aller vorhandenen NP-Strukturen

Die interne Struktur von Phrasen lässt sich zum Beispiel mit Hilfe von Strukturbäumen darstellen [WOE97].

Für das vorliegend Beispiel ergeben sich folgende Werte in Bezug auf die interne Struktur:

Interne Struktur NPs:

Recall = 159/159 entspricht einem Wert von 1

Precision = 163/273 entspricht ca. 0,6

Interne Struktur PPs:

Recall = 52/52 entspricht einem Wert von 1

Precision = 52/87 entspricht ca. 0,6

Auffällig ist, dass in Bezug auf die interne Struktur die Werte sowohl für die Nominal- als auch für die Präpositionalphrasen gleich sind.

Um die Arbeitsweise von SMES noch einmal genauer zu verdeutlichen wird anhand der Raketengeschichte 4 die komplette Evaluation beziehungsweise ihre Ergebnisse einmal detaillierter dargestellt:

Raketengeschichte 4 besteht in der originalen Fassung aus insgesamt 5 Sätzen. Sie beinhaltet 27 Tokens, was sich sehr schön an der Ausgabe von SMES ablesen lässt. Insgesamt wurden 4 der Sätze von SMES verarbeitet. Die Ausgabe von SMES sieht folgendermaßen aus:

```
File created on time(h:m:s): 16:13:49 date(d.m.y): 11.12.2006 ;

Number of tokens in text: 27

Total processing time for text (real time in msec): 0

Total processing time for token (real time in msec): 0
```

[0 *KERNSATZ* Heute fliegen ins Weltall . 0] [1 *KERNSATZ* Die Kind freuen sich sehr drauf ; 1] [2 *KERNSATZ* Jetzt ist es soweit . 2] [3 *ASYND-JUNKTION* Die Kind gehen in die Rakete rein und 109876543210 und 3] **Start die Kinder sind gestartet , Jetzt sind sie am Mond und fliegen wieder zurück .**

Dass von den fünf Sätzen nur 4 analysiert wurden, erkennt man an der Klammerung (0-3). Unterhalb dieser Klammerung, liefert SMES ein Glossar, ein Verzeichnis, in dem angelehnt an die Klammerung, eine genauere Analyse erfolgt:

GLOSSARY

Expr. 0:
```
[SENT-MARKER ".",
VF [SEM [SUBTYPE 'DATE-ADV,
...........MPRED "heute"]] & DATE-EXPR,
MF [SEM [HEAD <"ins">,
...........COMP [HEAD <"weltall">]]] & PP,
VERB [STEM "flieg",
.......FORM "fliegen"] & VERB] & KERNSATZ
```

Expr. 1:
```
[SENT-MARKER ";",
VF " Die",
....[SEM [HEAD <"kind">]] & NP,
MF [SEM [HEAD <"sich">]] & NP" sehr drauf",
VERB [STEM "freu",
.......FORM "freuen"] & VERB] & KERNSATZ
```

Expr. 2:
```
[SENT-MARKER ".",
VF " Jetzt",
MF [SEM [HEAD <"es">]] & NP" soweit",
VERB [STEM "sei",
.......FORM "ist"] & VERB] & KERNSATZ
```

Expr. 3:
```
[SENT-MARKER ".",
CONJUNCTS <[CONJUNCTS <[VF " Die",
.............................[SEM [HEAD <"kind">]] & NP,
.........................MF [SEM [HEAD <"in">,
.....................................COMP [QUANTIFIER <"d-det">,
.............................................HEAD <"rakete">]]] & PP"
rein und 109876543210",
.........................VERB [STEM "geh",
.............................FORM "gehen"] & VERB] & KERNSATZ,
.........................[VF [SEM [HEAD <"start">]] & NP[SEM [HEAD
<"kind">,
.....................................QUANTIFIER <"d-det">]] & NP,
.........................VERB [END2 15,
.............................START2 14,
.............................STEM "start",
.............................FORM "sind gestartet"] & VERB] &
KERNSATZ>,
............COOR "und",
............ART 'COMPLETE] & COORD-SATZ,
............[CONJUNCTS <[VF " Jetzt",
.........................MF [SEM [HEAD <"sie">]] & NP[SEM [HEAD
<"am">,
.....................................COMP [HEAD <"mond">]]] & PP,
.........................VERB [STEM "sei",
.............................FORM "sind"] & VERB] & KERNSATZ,
.........................[MF " wieder zurück",
.........................VERB [STEM "flieg",
.............................FORM "fliegen"] & VERB] &
STIRNSATZ>,
```

```
.............COOR "und",
.............ART 'INCOMPLETE] & COORD-SATZ>] & ASYND-JUNKTION
```

Diese Ausgabe wurde erzeugt durch die Option die Ausgabe in eine HTML-Datei umzuleiten. Man sieht, dass SMES über die Information der Phrasenerkennung hinaus auch Informationen wie Datumsangaben und ähnliches liefert. Anhand des Ausdrucks, der als Expr.2 gekennzeichnet ist, soll nun die Ausgabe genauer erläutert werden.

Expr. 2:

```
[SENT-MARKER ".",
VF " Jetzt",
MF [SEM [HEAD <"es">]] & NP" soweit",
VERB [STEM "sei",
.......FORM "ist"] & VERB] & KERNSATZ
```

Der Satzendepunkt wird als solcher gekennzeichnet (sentence-marker). SMES nutzt die Tatsache, dass eine Verbgruppe im Deutschen in einen rechten und linken Verbteil aufgespalten werden kann, für die Unterteilung eines Satzes. Das Vorfeld (VF= front field) steht für den linken Verbteil. Das Mittelfeld (MF= middle field) steht für den rechten Verbteil. VERB bezeichnet die Verbgruppe. Die implementierte Verbgrammatik erkennt jede einzelne Form von Verbvorkommen. Das HEAD-Element definiert die obere Grenze.
Die endgültige Ausgabe des Parsers für einen Satz ist eine „unterspezifizierte Dependenzstruktur“ (underspecified dependency structure = UDS). Eine UDS ist eine flache abhängigkeits-basierte Struktur eines Satzes, in der nur die oberen Grenzen für das Anhängen und das Scoping der Modifikatoren dargestellt wird [NEU00].

4.3.2 Fehleranalyse

Im Folgenden werden die Formen und Ursachen der in der Evaluation aufgetretenen Fehler beschrieben. Wo liegt die Verantwortlichkeit für inkorrekte Ergebnisse? Anschließend werden verschiedene Änderungen, die die Ergebnisse verbessern sollen, vorgeschlagen und diskutiert. Dabei soll unter anderem herausgearbeitet werden, wie komplex die einzelnen möglichen Erweiterungen tatsächlich sind/wären.

Die Verarbeitungsrichtung innerhalb des Shallow Text Processors von SMES ist immer bottom-up. Das bedeutet, dass nach der Verarbeitung durch den Scanner und die Analyse von Morphix, die verschiedenen Fragmenterkenner eingesetzt werden, um lokale Ausdrücke zu erkennen [NEU00]. Dadurch ergeben sich auch Probleme. Die Leistung des Satzparsers ist nach vorangegangener Definition der Arbeitsweise von SMES in großem Maße abhängig von der Korrektheit der vorgeschalteten Fragmenterkenner.

Dieses Problem verschärft sich unter anderem dann, wenn verschiedene Eigennamen oder andere spezifische Ausdrücke auftreten. Dieses Problem lässt sich jedoch gerade im vorliegenden Kontext einfach lösen. Durch die Vorgabe der Bilder und der Schreibhilfen sowie einzelner bereits vorgegebener Schlagwörter, lassen sich je nach Geschichte schon vorab verschiedene Eigennamen und andere spezifischen Ausdrücke einfach in das Lexikon einfügen. Dadurch erspart man dem Parser bereits den Umgang mit einigen unbekannten Wörtern. Selbstverständlich lassen sich nicht nur Eigennamen oder spezifische Ausdrücke (wie zum Beispiel Rakete oder Planet) in das Lexikon einfügen. Die Entscheidung darüber kann spontan und je nach Bedarf getroffen werden, da dieser Eingriff keine komplizierte oder ungerechtfertigt zeitintensive Modifikation darstellt.

Fehler und Probleme traten auch in Zusammenhang mit Koordinationen auf. In der Koordination werden Sätze durch Konjunktionen oder durch Satzzeichen voneinander getrennt. Dies tritt besonders dann auf, wenn einfach und wahrhaftig erzählt wird [WIK04]. Dies trifft für die Schreibweise der Kinder zu. Traten zum Beispiel mehrere Koordinationen innerhalb eines Satzes auf, kam es unter anderem zu Fehlern bei der Unterordnung der Nebensätze unter den Hauptsatz.

Ein weitere Fehler im Zusammenhang mit Koordinationen bestand darin, das es, wenn auch nur vereinzelt, dazu kam, dass eine nicht vorhandene Koordination von SMES erkannt wurde. Ergänzend sei hier angeführt, dass die Erkennung von Relativsätzen deutlich schwieriger ist als zum Beispiel die Erkennung von Interrogativsätzen. Die spielt jedoch in diesem Kontext eine untergeordnete Rolle, da die Kinder keinen Gebrauch von Relativsätzen machten. Dies ist ein Grund für die Eignung von SMES für die Verarbeitung der Schulaufsätze.

4.3.3 Evaluation der Arbeitsweise von SMES nach vorgenommenen Änderungen

Im diesem Kapitel soll die Frage erörtert werden, was wäre, wenn die Schnittstelle (SMES) in Bezug auf die Verarbeitung der Kindergeschichten besser wäre. Es geht hierbei um einfache Verbesserungen, die für den vorliegenden Korpus geeignet erscheinen. Dazu werden passende Heuristiken erstellt und anschließend die Ergebnisse betrachtet. Eine spezielle Frage die sich in diesem Kontext stellt, ist, ob es möglich ist, die Ergebnisse so zu modifizieren, dass auf die Schreibkompetenz der Kinder Rücksicht genommen wird.

Manuelles Einfügen von Eigennamen und Berufsbezeichnungen in das Lexikon

Wie bereits erwähnt ist dies die am einfachsten zu begründende und auch die am einfachsten durchzuführende Modifikation. Vorausgesetzt wird ein fehlerfreies Einfügen in das Lexikon. Das Ergebnis lässt sich aber gerade durch diese simpel erscheinende Ergänzung im Kontext der Aufgabenstellung, die die Schulkinder bekommen, deutlich verbessern. In der vorliegenden Geschichte empfiehlt es sich die Eigennamen wie Anja und Carlotta in die Liste der Namen mit auf zu nehmen. Je nach Kompetenzziel könnte man sogar erwägen, diese Namen nicht nur in der korrekten Schreibweise in das Lexikon zu übertragen, sondern in jeder Schreibweise, die die Kinder in ihren Geschichten benutzen. Dadurch ermöglicht man dem Parser zum Beispiel eine größere Chance zur Phrasenerkennung, da er nicht mehr in einem solch hohen Maß wie zuvor von dem vorangegangenen Erkenner für Eigennamen negativ beeinflusst wird.

Annahme von Sätzen einer festen Länge/Automatisches Satzende setzen

Ein Fehler, den die Kinder nicht nur häufig, sondern auch ausnahmslos und durchgehend machten, war die fehlende oder falsche Interpunktion. Die Kinder setzten Satzzeichen in der Regel sehr selten und wenn, dann meist auch sehr willkürlich. Deshalb ist eine Idee für eine Modifikation, Sätze einer bestimmten Länge anzunehmen. Damit soll dem Parser die Möglichkeit geben werden, nicht mehr mit Sätzen arbeiten zu müssen, die sich über mehrere Zeilen erstrecken und damit das Ergebnis zu verbessern. Dazu wurden einmal Sätze einer festen Länge von 3 Wörtern und, zum Vergleich, Sätze einer festen Länge von 5 Wörtern angenommen.

Genauer gesagt heißt das, dass automatisch nach der jeweils genannten Anzahl von Wörtern automatisch ein Satzendezeichen, in diesem Fall ein Punkt, gesetzt wird. Die einzige Einschränkung, die in diesem Zusammenhang gemacht wurde, war, dass keine Satztrennung stattfinden sollte, wenn das letzte Wort ein Artikel war oder eine Präposition war. Nachdem automatisch, von dieser Heuristik ausgehend, nach jeweils 3 oder 5 Wörtern ein Punkt gesetzt wurde, wurde damit automatisch der Beginn einer neuen Zeile festgelegt (damit das notwendige Eingabeformat erhalten bleibt) und in der nächsten Zeile wurde mit einem Großbuchstaben begonnen. Das Resultat dieser Modifikation wird wieder anhand der Raketengeschichte 4 verdeutlicht. Das ursprüngliche Ergebnis ohne Modifikationen ist uns bereits bekannt.

Nach automatischem Setzen eines Satzendes nach 3 Wörtern liefert SMES dieses Ergebnis:

```
File created on time(h:m:s): 8:10:42 date(d.m.y): 6.2.2008 ;

Number of tokens in text: 51

Total processing time for text (real time in msec): 0

Total processing time for token (real time in msec): 0

Heute fliegen ins Weltall . Die Kind freuen . Sich sehr drauf . Jetzt ist es soweit . Die
Kind gehen . In die Rakete . Rein und 109876543210 . Und Start die Kinder . Sind
gestartet , Jetzt . Sind sie am Mond . Und fliegen wieder . Zurück .
```

Abb. 4.3.1 Ergebnis für automatisches Satzende nach 3 Wörtern

SMES konnte die Eingabe nicht verarbeiten. Das entsprechende Glossar bleibt demzufolge leer.

Nach automatischem Satzende nach 5 Wörtern ist dies das Ergebnis:

File created on time(h:m:s): 8:19:37 date(d.m.y): 6.2.2008 ;

Number of tokens in text: **48**

Total processing time for text (real time in msec): **0**

Total processing time for token (real time in msec): **0**

Heute fliegen ins Weltall . Die Kind freuen sich sehr . Drauf ; Jetzt ist es soweit . Die Kind gehen in die Rakete . Rein und 109876543210 und Start . [0 SIMPLE Die Kinder sind gestartet , Jetzt . 0] Sind sie am Mond und . Fliegen wieder zurück .

Abb. 4.3.2 Ergebnis für automatisches Satzende nach 5 Wörtern

Das entsprechende Glossar sieht so aus:

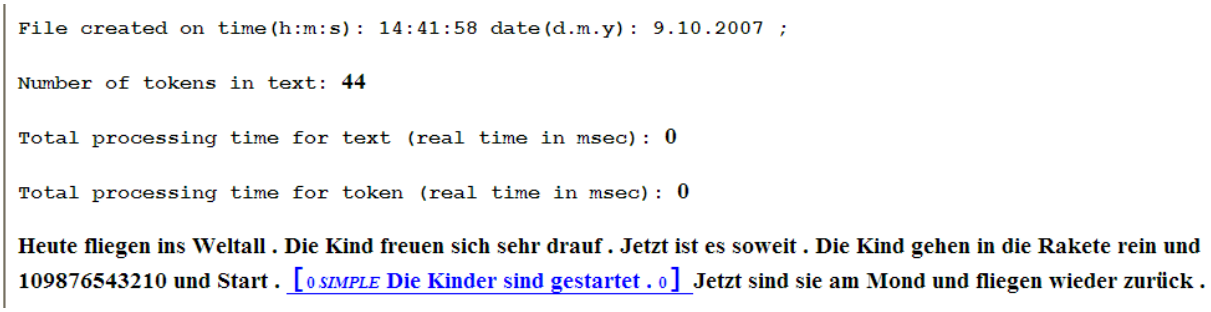
```
Expr. 0:
[COMPLETE 'YES,
SYN [SUBJ [RANGE [SEM [HEAD <"kind">,
.........................QUANTIFIER <"d-det">]] & NP],
......NP-MODS <>,
......PP-MODS <>,
......SC-MODS <>,
......VCS-MODS <>,
......PROCESS [END2 4,
...............START2 3,
...............STEM "start",
...............FORM "sind gestartet"] & VERB,
......SQL-TYPE 'GF-VERB-NODE] & SUBJ] & SIMPLE
```

Abb. 4.3.3 Glossar zum automatischen Satzende nach 5 Wörtern

Satzzeichen setzen

Im direkten Vergleich dazu wurde kein automatisches Satzende gesetzt, sondern es wurden alle Satzzeichen von Hand so verbessert, wie es im gegeben Kontext korrekt gewesen wäre. Die Hoffnung in diesem Ansatz besteht darin, dass der Parser mit den korrekten Satzzeichen trotz der Rechtschreibfehler und der Wortstellungsfehler besser in der Lage ist Sätze zu erkennen und damit das Ergebnis besser ausfällt als zuvor.

Diese Modifikation angewandt auf die Geschichte 4 liefert die folgenden Ergebnisse:



File created on time(h:m:s): 14:41:58 date(d.m.y): 9.10.2007 ;

Number of tokens in text: **44**

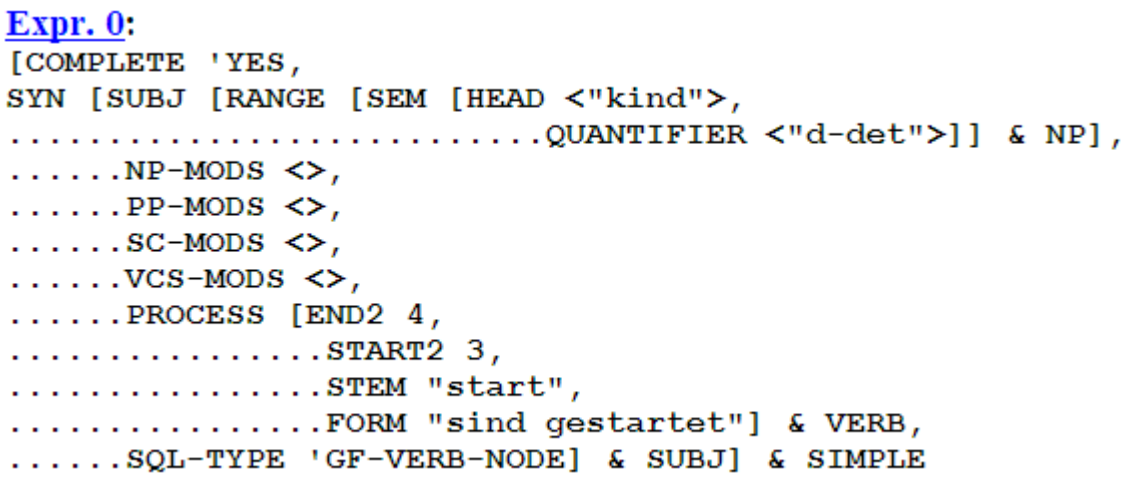
Total processing time for text (real time in msec): **0**

Total processing time for token (real time in msec): **0**

Heute fliegen ins Weltall . Die Kind freuen sich sehr drauf . Jetzt ist es soweit . Die Kind gehen in die Rakete rein und 109876543210 und Start . [0 *SIMPLE* Die Kinder sind gestartet . 0] Jetzt sind sie am Mond und fliegen wieder zurück .

Abb. 4.3.4 Geschichte 4 mit korrekten Satzzeichen

Das entsprechende Glossar:



```
Expr. 0:
[COMPLETE 'YES,
SYN [SUBJ [RANGE [SEM [HEAD <"kind">,
.........................QUANTIFIER <"d-det">]] & NP],
......NP-MODS <>,
......PP-MODS <>,
......SC-MODS <>,
......VCS-MODS <>,
......PROCESS [END2 4,
................START2 3,
................STEM "start",
................FORM "sind gestartet"] & VERB,
......SQL-TYPE 'GF-VERB-NODE] & SUBJ] & SIMPLE
```

Abb. 4.3.5 Glossar zu Geschichte 4, korrekte Satzzeichen

5 Zusammenfassung und Ausblick

Ziel dieser Arbeit war es, die Möglichkeiten des praktischen Einsatzes des Computers, in Bezug auf die spezielle Textart, die transliterierten Aufsätze der Schulkinder, die innerhalb eines Projektes mit einem konkreten Ziel, der Unterstützung und Verbesserung des Unterrichts, entstanden, zu erkunden. Es galt eine geeignete linguistische Methode zu finden, die in der Lage ist, die spezielle Textform so tief wie möglich zu verarbeiten. Darüber hinaus sollte außerdem erarbeitet werden, welche Möglichkeiten sich in dem gegebenen Rahmen bieten, die erzielten Ergebnisse durch Modifikationen entsprechend zu verbessern. Dazu wurden im Vorfeld verschiedene Methoden und Werkzeuge der Linguistik beleuchtet. Aufgrund der vorliegenden Problemstellung wurde dabei ausschliesslich der syntaktische Aspekt berücksichtigt. Der Einsatz eines flachen, partiellen Parsers an dieser Stelle stellt einen Weg dar, der diesen Anforderungen genügt. Für das Problem der Erkennung von Komposita beispielsweise, muss keine vollständige syntaktische Analyse vorgenommen werden. Ziel war keine vollständige, tiefe Analyse, sondern die Wahl einer Methode, die in der Lage ist, auch und gerade mit einem schlecht geformten Input umzugehen. Deshalb fiel die Wahl auf flache Parser. Speziell Chunk Parsing erwies sich als attraktive Möglichkeit der Verarbeitung der Geschichten der Schulkinder. Die Qualität des Inputs war verhältnismäßig niedrig. Wiederholte oder fehlende Wörter, syntaktische Fehler oder Schreibfehler können aber genau in diesem Parsingansatz gut kompensiert werden. Mit seiner strikten bottom-up Parsingstrategie, bei der einfache, nicht rekursive Phrasen erkannt und dann zu komplexen Einheiten kombiniert werden, bietet SMES eine gute und Grundlage für die Verarbeitung der Geschichten. SMES hat sich vor allem dadurch als sinnvoll einsetzbar erwiesen, dass die Modifikationen, die zu einem besseren Ergebnis führen sollen, in dem gegebene Kontext einfach und effizient umsetzbar sind. Durch die Art und Weise der Aufgabenstellung, die den Kindern im Rahmen des VERA-Projektes vorliegen, bietet es sich an, auf ein System wie SMES zurück zu greifen. Durch eine einfache und legitime Ergänzung des Lexikons, die sich durch die Vorgabe von Kontext der Geschichte, vorgegebene Bilder und Wortvorschläge bereits anbietet, kann schon von im Vorfeld eine gute Ausgangsposition für die Arbeit des Parsers geschaffen werden. Denn indem man jene Bedeutungen, die in einer bestimmten Domäne unwahrscheinlich sind, ausschließt, beziehungsweise umgekehrt, wie beschrieben, jene Bedeutungen, mit denen mit großer Wahrscheinlichkeit zu rechnen ist, direkt integriert, verringert man automatisch den Disambiguierungsaufwand.

Die grundlegende Frage, die sich in einem solchen Kontext immer stellt ist: „Sind die gegebenen Beschränkungen des Parsers für die konkrete Aufgabenstellung akzeptabel?“

In Bezug auf SMES und die Analyse von transliterierten Schulaufsätzen kann man diese Frage wohl grundsätzlich positiv beantworten. Man kann bei einer Evaluation verschiedene Korrektheitskriterien nutzen. Das bedeutet, dass man selbst festlegen kann, welche Werte innerhalb eines Ausdrucks übereinstimmen müssen, um den Ausdruck, im festgelegten Sinn, korrekt sein zu lassen.

Auch wenn das System, so wie es ursprünglich geplant und entwickelt wurde, ohne geeignete Modifikationen nicht per se die optimale Verarbeitungsmethode darstellt, ist es jedoch erst einmal flexibel und robust genug mit der Art von Daten, wie sie gegeben sind, zu arbeiten und brauchbare Ergebnisse für eine Weiterverarbeitung zu liefern.

Die Analyse der, während der Evaluationsstudie entstandenen Fehler, liefert einen Einblick in die, teilweise umfangreichen, Probleme, die sich beim flachen Parsen ergeben können. Bei der flachen Analyse werden nur die eindeutig interpretierbaren Konstituenten analysiert. Die Analyse kann deshalb lückenhaft bleiben. Die Verarbeitungsrichtung innerhalb des Shallow Text Processors von SMES ist immer streng bottom-up: nach der Vorverarbeitung durch Scanner und Morphix werden die verschiedenen Fragmenterkenner eingesetzt. Ein Problem, das sich bei dieser Analyseart stellt ist, dass man sich mit Fehlern, die durch vorgeschaltete Module entstanden sind, auseinandersetzen muss. Eine entsprechende Vorverarbeitung bietet Möglichkeiten, das Ergebnis zu verbessern. Man kann zum Beispiel eigenständig Transformationsregeln erstellen. Nach der Initialisierung eines Wortes mit einem Tag, kann man dann die Transformation wählen, die die Fehlerrate am stärksten senkt. Nachteil dieser Methode ist jedoch die mangelnde Robustheit und die Tatsache, dass man dazu ein handgetaggtes Korpus benötigt. Mögliche Erweiterungen, die fehlerhafte Erkennungen vermeiden könnten, sind von unterschiedlicher Komplexität.

Der Wunsch und das Bedürfnis nach Vergleichsmöglichkeiten, Unterstützung für die Lehrkräfte und die Notwendigkeit den Kindern die Interaktion mit dem Computer natürlich, benutzerfreundlich und effizient zu gestalten, wird immer größer.

Auch wenn man noch nicht in der Lage ist, das Ziel zu erreichen, mit zunehmender Güte der Systeme zur Hilfestellung und der damit verbundenen notwendigen Evaluation der vorhandenen Ansätze, die so aussagekräftig und idealerweise so weit übertragbar wie möglich sein sollte, rückt dieses Ziel immer näher.

6 Verzeichnisse

6.1 Literaturverzeichnis

ABN91 Steven Abney : Parsing by chunks, In: Robert Berwick, Steven Abney, Carol Tenny (Hrsg.), *Principle-Based Parsing,* Kluwer Academic Publishers, Dordrecht, 1991.

ABN96 Steven Abney: Partial Parsing via Finite-State Cascades, Journal of Natural Language Engineering, 1996, S. 337-344.

AHO06 Alfred Aho, Ravi Sethi, Jeffrey Ullman, Monica S. Lam, Pearson Education Inc , *Compilers:Principles Techniques, and Tools*, 2006.

ALS92 Hiyan Alshawi: The Core Language Engine, The MIT Press, ISBN:0585033552 Cambridge/Massachusetts, 1992.

AND05 Maria Andris, Elena Frick, Eva Sourjikova: Part-Of-Speech-Tagging mit Transduktoren, Hauptseminar „Endliche Automaten für die Sprachverarbeitung“, Universität Heidelberg, 2005.

ASC05 Bianca Aschenberger, Petra Wagner: Diagnostische Evaluation der Sprachapplikation von GPS-Navigationssystemen, IKP-Arbeitsbericht NF 15, Universität Bonn, 2005.

BAU99 Peter Baumgärtner: Evaluation mediengestützten Lernens: Theorie-Logik-Modelle . In: Michael Kindt (Hrsg.): *Projektevaluation in der Lehre Multimedia an Hochschulen zeigt Profil(e)*, Münster, Waxmann, 1999, S. 71.

BES05 Markus Bestehorn: Part of Speech Tagging, In: Text Mining: Wissensgewinnung aus natürlichsprachlichen Dokumenten, Interner Bericht 2006-5, S.59-82, Universität Karlsruhe, Fakultät für Informatik, Institut für Programmstrukturen und Datenorganisation (IPD), 2006. ISSN:1432-7864.

BIG03 Johnny Bigert, Ola Knutsson, Jonas Sjöbergh: Automatic evaluation of robustness and degradation in tagging and parsing, Borovets, Bulgarien, 2003.

BIG05 Johnny Bigert, Jonas Sjöbergh, Ola Knutsson, Magnus Sahlgren : Automatic evaluation of parser robustness: Eliminating manual labor and annotated resources, Mexico City, 2005, S. 142-154.

BLA93 Ezra Black, Roger Garside, Georey Leech : Statistically-driven Computer Grammars of English: The IBM/Lancaster Approach, Amsterdam, 1993.

BRA00 Thorsten Brants: TnT- A statistical Part-Of-Speech-Tagger, In *6th Applied Natural Language Processing (ANLP '00), April 29 - May 4*, pages 224-231, Seattle, USA, 2000. Association for Computational Lingusitics.

BRA02 Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, George Smith: The TIGER Treebank in Proceedings of the Workshop on Treebanks and Linguistic Theories, Sozopol, 2002.

BRI87 Ted Briscoe, Claire Grover, Bran Boguraev, John Carroll : Feature defaults, propagation and reentrancy, In: E. Klein und J. van Bentham (Hrsg.): *Categories, Polymorphism and Unification*, Centre of Cognitive Science, University of Edinburgh, 1987, S. 19-34.

BRI92 Eric Brill: A simple rule-based part-of-speech tagger, ACL, Trento/Italien, 1992, S. 152-155.

BRI94 Eric Brill: A report of recent progress in transformation-based error-driven learning, Proceedings of AAAI, 1994.

BRI95 Ted Briscoe, John Carroll: Developing and evaluating a probabilistic LR parser of part-of-speech and punctuation labels, Prag/Tschechien, 1995, S. 48-58.

BUß83 Hadumod Bußmann: Lexikon der Sprachwissenschaft, Stuttgart: Kröner Verlag, 1983 (2.Auflage 1990).

CAR04 Kai-Uwe Carstensen, Christian Ebert, Cornelia Endriss, Susanne Jekat, Ralf Klabunde: Computerlinguistik und Sprachtechnologie. Eine Einführung, Heidelberg: Spektrum-Verlag, 2004.

CAR98 John Carroll, Ted Briscoe, Antonio Sanfilippo: Parser Evaluation: a Survey and a New Proposal, In Proceedings 1st Conference on Linguistic Resources, Granada/Spanien, 1998, S. 447-454.

CHR02 Gerd Christoph, Horst Hackel: Starthilfe Stochastik, Stuttgart *Leipzig* Wiesbaden: B.G. Teubner Verlag, ISBN:3519003414, 2002.

DIT03 Ullrich Dittler: E-Learning Einsatzkonzepte und Erfolgsfaktoren des Lernens mit interaktiven Medien, R. Oldenbourg Verlag, München/Wien, ISBN 3486258079, 2003.

EIB02 Maximilian Eibl: Evaluation, Vorlesung Medientools, TU Chemnitz, www.tu-chemnitz.de/.../lehre/lehrmaterial/skripte/medienwerkzeuge/medientools_eibl_05_evaluation_1.ppt, 2002.

EUL04 Dieter Euler et al.: E-Learning in Hochschulen und Bildungszentren , In: Dieter Euler (Hrsg.), R. Oldenbourg Verlag, ISBN 3-486-20008-9, München/Wien, 2004.

FAN93 Gisbert Fanselow, Sascha W. Felix: Sprachtheorie: eine Einführung in die generative Grammatik, Tübingen, 2 Bände, 1993.

FUH06 Norbert Fuhr: Information Retrieval, Vorlesung Universität Duisburg-Essen, 2006.

GAL93 Julia R. Galliers, Karen Sparck Jones: Evaluating natural language processing systems , In: Computer Laboratory, 1993.

GOO02 Gerhard Goos: Vorlesungen über Informatik Band 1, Grundlagen und funktionales Programmieren, Springer Verlag Heidelberg, 2002.

GRE71 Barbara B. Greene, Gerald M. Rubin: Automatic grammatical tagging of English Technical report, Providence/Rhode Island, 1971.

HAR91 Phil Harrison, Steven Abney, Ezra Black, Dan Flickinger, Claudia Gdaniec, Ralf Grishman, Don Hindle, Bob Ingria, Mitch Marcus, Beatrice Santorini, Tomek Strzalkowski: Evaluating syntax performance of parser/grammars of English, ACL, 1991.

HER04 Jürgen Hermes: Part of Speech Tagging (1), Universität Köln, Institut für Linguistik-Sprachliche Informationsverarbeitung, http://www.spinfo.uni-koeln.de/default/live/lehre/QLKurs/QLMaterial/QL_16.pdf, 2004.

HES06 Michael Hess: Chunk Parsing, http://www.ifi.uzh.ch/CL/hess/classes/le/chunk.0.l.pdf, Universität Zürich, Institut für Computerlinguistik, 2006.

HOB96 Jerry R. Hobbs, Douglas Appelt, John Bear,David Israel, Megumi Kameyama, Mark Stickel, Mabry Tyson : FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text, Roche und Schabes, Cambridge/Massachusetts, 1996.

JUR00 Daniel Jurafsky, James H. Martin: Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Prentice-Hall, 2000.

KER88 Brian Kernighan, Dennis Ritchie: The C Programming Language, 2nd Edition, Prentice Hall, 1988.

KLE03 Ursula Klenk: Grammar, comparative and general, Gunter Narr Verlag, 2003.

LEE83 Geoffrey Leech, Roger Garside, Eric Atwell: The Automatic Grammatical Tagging of the LOB Corpus, 1983, S. 13-33.

LIN95 Dekang Lin: A dependency-based method for evaluating broad-coverage parsers, Montreal/Kanada, 1995, S. 1420-1425.

MAN00 Christopher D. Manning, Hinrich Schuetze: Foundations of Statistical Natural Language Processing, MIT Press, Cambridge/Massachusetts, 2000.

MAR82 David Marr: Vision. A computational investigation into the human representation and processing of visual information, In:W.H. Freeman (Hrsg.), New York, 1982.

MEN06 Wolfgang Menzel: Sprachverarbeitung-Ein Überblick, In:Handbuch der künstlichen Intelligenz, G. Görz, C.-R. Rollinger, J. Schneeberger (Hrsg.), Kapitel 16, Oldenbourg Verlag,2006.

MEY07 Meyers Lexikon, Bibliographisches Institut & F. A. Brockhaus AG, Meyers Lexikonverlag, 2007.

MIT98 Hartmut Mitzlaff et al.: Grundschule und neue Medien. In: Hartmut Mitzlaff, Angelike Speck-Hamdan (Hrsg.), Frankfurt am Main, 1998.

MOL03 Diego Molla, Ben Hutchinson: Intrinsic versus Extrinsic Evaluations of Parsing Systems, Budapest, 2003, S. 43-50.

MUE06 Frank Henrik Müller: Wortarten-Tagging, Institut für deutsche Sprache und Linguistik, Humboldt Universität Berlin, 2006.

NEU00 Günther Neumann, Christian Braun, J. Piskorski: A Divide-and-Conquer Strategy for Shallow Parsing of German Free Texts, Seattle/Washington, 2000, S. 239-246.

NEU06 Günther Neumann, http://www.dfki.de/~neumann/pd-smes/pd-smes.html, LT-lab, DFKI, Saarbrücken, 2006.

NEU97 Günther Neumann, Rolf Backofen, Judith Baur, M. Becker, Christian Braun: An information extraction core system for real world german text processing, Washington/ USA, 1997, S. 208-215.

POL87 Carl Pollard, Ivan Sag: Information-based Syntax and Semantics. Volume 1: Fundamentals, Chicago, 1987.

POL94 Carl Pollard, Ivan Sag (University of Chicago Press): Head-Driven Phrase Structure Grammar, Chicago, 1994.

SCH02 Karl-Michael Schneider: Parsingformalismen, http://www.phil.uni-passau.de/linguistik/lehre/parsingformalismen/, Universität Passau, 2002.

SCH95 Anne Schiller, Simone Teufel: Guidelines für das Tagging deutscher Textcorpora mit STTS, Technical Report, IMS-CL, Universität Stuttgart, 1995.

STO04 Karlheinz Stöber, Bernhard Schröder, Wolfgang Hess: Vom Text zur gesprochenen Sprache. In: Henning Lobin, Lothar Lemnitzer (Hrsg.): *Texttechnologie, Perspektiven und Anwendungen*, Tübingen: Stauffenburg, 2004, S. 295-325.

TEI07 Christoph Teichmann: Parser-Evaluierung, http://www.ling.uni-potsdam.de/~jonask/teaching/Parsing-SS07/teichmann.pdf, Universität Potsdam, 2007.

TRE06 Galina Trefilova, Susanne O`Shaughnessy: Sprachmodelle Hauptseminar Dialogsysteme, http://www.cis.uni-muenchen.de/people/kristof/DIALOG05/sitzung10a.pdf, Universität München, 2006.

VER06 Vera-Vergleichsarbeiten in der Grundschule, www.uni-landau.de/vera, 2006.

WAR97 Dor Warth: Künstliche Intelligenz: Spracherkennung und Sprachverstehen, Referat zum Hauptseminar "Psycholinguistik: Mentale Prozesse in der Sprachverarbeitung", 1997.

WIK02 Pragmatik (Lingusitik), In: Wikipedia Die freie Enzyklopädie, www.wikipedia.de.

WIK04 Parataxe, In: Wikipedia Die freie Enzyklopädie, www.wikipedia.de.

WOE97 Angelika Wöllstein-Leisten, Axel Heilmann, Peter Stephan, Sten Vikner: Deutsche Satzstruktur. Grundlagen der syntaktischen Analyse, ISBN : 978-3-86057-272-6, Tübingen: Stauffenburg Verlag, 1997.

WOM89 Prof. Dr. Christa Womser-Hacker: Der PADOK-Retrievaltest Zur Methode und Verwendung statistischer Verfahren bei der Bewertung von Information-Retrieval-Systemen, Regensburg, 1989, S. 34.

6.2 Abbildungsverzeichnis

7 Anhang

POS =	Beschreibung	Beispiele
ADJA	attributives Adjektiv	*[das] große [Haus]*
ADJD	adverbiales oder prädikatives Adjektiv	*[er fährt] schnell* *[er ist] schnell*
ADV	Adverb	*schon, bald, doch*
APPR	Präposition; Zirkumposition links	*in [der Stadt], ohne [mich]*
APPRART	Präposition mit Artikel	*im [Haus], zur [Sache]*
APPO	Postposition	*[ihm] zufolge, [der Sache] wegen*
APZR	Zirkumposition rechts	*[von jetzt] an*
ART	bestimmter oder unbestimmter Artikel	*der, die, das,* *ein, eine*
CARD	Kardinalzahl	*zwei [Männer], [im Jahre] 1994*
FM	Fremdsprachliches Material	*[Er hat das mit "]* *A big fish [" übersetzt]*
ITJ	Interjektion	*mhm, ach, tja*
KOUI	unterordnende Konjunktion mit "zu" und Infinitiv	*um [zu leben],* *anstatt [zu fragen]*
KOUS	unterordnende Konjunktion mit Satz	*weil, daß, damit,* *wenn, ob*
KON	nebenordnende Konjunktion	*und, oder, aber*
KOKOM	Vergleichspartikel, ohne Satz	*als, wie*
NN	Appellativa	*Tisch, Herr, [das] Reisen*
NE	Eigennamen	*Hans, Hamburg, HSV*
PDS	substituierendes Demonstrativ–pronomen	*dieser, jener*
PDAT	attribuierendes Demonstrativ–pronomen	*jener [Mensch]*
PIS	substituierendes Indefinit–pronomen	*keiner, viele, man, niemand*
PIAT	attribuierendes Indefinit–pronomen ohne Determiner	*kein [Mensch],* *irgendein [Glas]*
PIDAT	attribuierendes Indefinit–pronomen mit Determiner	*[ein] wenig [Wasser],* *[die] beiden [Brüder]*
PPER	irreflexives Personalpronomen	*ich, er, ihm, mich, dir*
PPOSS	substituierendes Possessiv–pronomen	*meins, deiner*
PPOSAT	attribuierendes Possessivpronomen	*mein [Buch], deine [Mutter]*
PRELS	substituierendes Relativpronomen	*[der Hund,] der*

Abb. 7.1 Tabelle STTS Teil1

POS =	Beschreibung	Beispiele
PRELAT	attribuierendes Relativpronomen Relativpronomen	*[der Mann ,] dessen [Hund]*
PRF	reflexives Personalpronomen	*sich, einander, dich, mir*
PWS	substituierendes Interrogativpronomen	*wer, was*
PWAT	attribuierendes Interrogativpronomen	*welche [Farbe], wessen [Hut]*
PWAV	adverbiales Interrogativ– oder Relativpronomen	*warum, wo, wann, worüber, wobei*
PAV	Pronominaladverb	*dafür, dabei, deswegen, trotzdem*
PTKZU	"zu" vor Infinitiv	*zu [gehen]*
PTKNEG	Negationspartikel	*nicht*
PTKVZ	abgetrennter Verbzusatz	*[er kommt] an, [er fährt] rad*
PTKANT	Antwortpartikel	*ja, nein, danke, bitte*
PTKA	Partikel bei Adjektiv oder Adverb	*am [schönsten], zu [schnell]*
TRUNC	Kompositions–Erstglied	*An– [und Abreise]*
VVFIN	finites Verb, voll	*[du] gehst, [wir] kommen [an]*
VVIMP	Imperativ, voll	*komm [!]*
VVINF	Infinitiv, voll	*gehen, ankommen*
VVIZU	Infinitiv mit "zu", voll	*anzukommen, loszulassen*
VVPP	Partizip Perfekt, voll	*gegangen, angekommen*
VAFIN	finites Verb, aux	*[du] bist, [wir] werden*
VAIMP	Imperativ, aux	*sei [ruhig !]*
VAINF	Infinitiv, aux	*werden, sein*
VAPP	Partizip Perfekt, aux	*gewesen*
VMFIN	finites Verb, modal	*dürfen*
VMINF	Infinitiv, modal	*wollen*
VMPP	Partizip Perfekt, modal	*[er hat] gekonnt*
XY	Nichtwort, Sonderzeichen enthaltend	*D2XW3*
$,	Komma	,
$.	Satzbeendende Interpunktion	. ? ! ; :
$(	sonstige Satzzeichen; satzintern	– [] ()

Abb. 7.2 Tabelle STTS Teil2