Università
Ca'Foscari
Venezia

Corso di Laurea magistrale
(*ordinamento ex D.M. 270/2004*)
in *Scienze del linguaggio*

Tesi di Laurea

Ca' Foscari
Dorsoduro 3246
30123 Venezia

GI-Tutor: ein deutscher Grammatikchecker für Lernende mit italienischer Muttersprache

Relatore
Ch. Prof. Rodolfo Delmonte

Correlatori
Prof. Roland Hinterhölzl
Dott. Peter Paschke

Laureando
Alessia Battisti
827875

Anno Accademico
2014 / 2015

Inhaltsverzeichnis

Inhaltsverzeichnis

1. Einleitung

Wie die Erfahrung lehrt, ist die Redewendung „Durch Fehler wird man klug“ nicht in jedem Bereich unbegründet.
Unterschiedliche Theorien der Fremdsprachenlehr- und -lernforschung haben verschiedene Ansichten zu den Sprachfehlern, die ein Lernender machen kann. Der Behaviorismus erklärt, dass Fehler unbedingt vermieden werden sollten - zum Beispiel beim „Überlernen“ (übertriebenes Lernen, engl. *overlearning*). Demgegenüber glauben die Theoretiker der *Interlanguage*, dass die Fehler Idiosynkrasien im L2-System des Lernenden sind, die nur mit Bezug auf die Zielsprache behandelt werden sollen. Die Forscher des Kognitivismus meinen, dass der Fokus des Lernenden auf die richtige Form und Struktur des zu Lernenden das Erlernen aktivieren kann.

In der Fremdsprachenlehr- und -lernforschung haben die Computer eine erhebliche Rolle gespielt, weil sie den Lernenden verschiedene Typologien von Sprachübungen und -aufgaben anbieten können und dazu eine unmittelbare Rückmeldung (Feedback) erzeugen. Nach Dodigovic (2005) sind diese intelligenten computergestützten Systeme für Fehlererkennung und -diagnose in der schriftlichen Sprache ziemlich hochentwickelt und verwenden spezielle Teilsysteme und Module, die mit vielen verschiedenen Bearbeitungsniveaus und Ebenen der Sprache verbunden sind - wie bspw. der syntaktischen Ebene.
Diese Arbeit richtet deshalb große Aufmerksamkeit auf die Verwendung des Parsers: Mit dem Begriff Parser ist ein syntaktischer Analysator gemeint, der einen in natürlicher Sprache formulierten Satz identifiziert und analysiert, um festzustellen, ob der gegebene Satz der im Parser beinhalteten Grammatik entspricht. In Allgemeinen besteht ein Parser hauptsächlich aus zwei wichtigen Teilen: einer Grammatik, die die Regeln der Sprache enthält, und einem Parsing-Algorithmus, der diese Grammatik interpretiert und die Analyse durchführt. Die Verwendung eines Parsers erlaubt es somit, Sprache zu analysieren, Fehler zu erkennen und letztlich den Lernenden ein Feedback zu geben.
Wie Sennrich (2009) erwähnt, weckt die Analyse bzw. das Parsen des Deutschen Aufmerksamkeit, weil das Deutsche eine wichtige europäische Sprache ist, und es die besonderen Eigenschaften der freien Wortstellung und einer reichen Morphologie hat.

Daher konzentriert sich die vorliegende Arbeit auf zwei Parser, die zum Analysieren deutscher Sätze entwickelt wurden.

Nach einer Einführung in den Zusammenhang zwischen der *Verarbeitung der natürlichen Sprache* (NLP) und *Computer-Assisted Language Learning (CALL)* (Kapitel 2), fokussiert sich diese Arbeit auf den Gebrauch von Parsern im Bereich des Sprachlehrens (Kapitel 3). Hierzu werden zwei verschiedene Grammatiksysteme zur automatischen Textanalyse für die deutsche Sprache betrachtet und dargestellt: Zuerst wird der Dependenz-Parser *ParZu* detailliert beschrieben; danach der an der Universität Ca' Foscari entwickelter *GI-Tutor*.
Der GI-Tutor ist ein Konstituenz-Grammatikchecker, der für Lernende mit italienischer Muttersprache gestaltet wurde. Seine Systemstruktur wurde einer vorherigen Studie entnommen und implementiert, um eine beträchtliche Anzahl von Sätzen, mit verschiedenen syntaktischen Strukturen, zu analysieren.

Zunächst wurde das Lexikon, das circa 8000 Einträge enthält, von Hand organisiert und strukturiert. Daraufhin dehnte eine angepasste Version des JWCDG Lexikons dieses manuell erstellte Lexikon weiter aus. Zudem bildeten fehlerhafte Sätze aus den Klausuren aus dem ersten Studienjahr des Bachelorstudiengangs „Lingue, culture e scienze del linguaggio“ an der Universität Ca' Foscari die Grundlage für ein Trainingskorpus.

Kapitel 4 stellt die grundlegenden Begriffe der Fehler und Fehleranalyse vor und erklärt danach die Klassifizierung der im Trainingskorpus gesammelten Fehler. Dies erlaubt eine kleine Statistik der Hauptproblematiken der Lernenden. Es wird darüber hinaus die Leistung des Konstituenz-Grammarcheckers bewertet, um die Typen und Häufigkeit der Fehler, die er erfolgreich erkennen kann, zu bestimmen. Zudem wird der GI-Tutor mit ParZu verglichen und die dadurch erhaltenen Ergebnisse veranschaulicht.
GI-Tutor möchte die Aufmerksamkeit der Lernenden auf die richtige Struktur eines Satzes richten. Daher zeigt das System eine Benutzeroberfläche, auf der dem Lernenden drei verschiedene Übungen angeboten werden. Diese Aufgaben werden dann durch einen Vergleich mit den Strukturen und Regeln der Parsergrammatik analysiert und korrigiert. Letztlich erzeugt das System ein Feedback, das eine explizite

Erklärung der Fehler, die der Lernende begeht, darstellt.

Letzten Endes schließt das Kapitel 5 mit Schlussfolgerungen und Fazit diese Arbeit ab.

1.1. Danksagungen

Die vorliegende Arbeit verdankt ihre Entstehung dem Interesse von Herr Prof. Dr. Rodolfo Delmonte, der bei der Vorbereitung und Durchführung der Untersuchung wertvolle und wesentliche Unterstützungsarbeit geleistet hat: Ihm gebührt mein Dank zuallererst.

Das Zustandekommen wäre nicht möglich gewesen, wenn Herr Prof. Dr. Hinterhölzl und Herr M.A. Paschke nicht bereit gewesen wären, mit mir und Herrn Delmonte zu kooperieren: Für ihre Unterstützung und Mitarbeit sei Ihnen an dieser Stelle sehr herzlich gedankt.

Nicht zuletzt gilt mein Dank auch all meinen Verwandten und Freunden, die mich in diesen Studienjahren immer ertragen und unterstützt haben. Im Besonderen bedanke ich mich vom ganzen Herzen bei meinen Eltern Fiorenza und Amos, meiner Schwester Ilenia, meinen Freunden Elisabetta, Annachiara, Eugeniu, Chiara, Luca und Marika, die mir vertraut und an meine Fähigkeiten geglaubt haben, und Jessi, die mir aus Deutschland seelisch bestand und bei sprachlichen Schwierigkeiten stets bereitwillig geholfen hat: Ich möchte euch allen meinen herzlichsten Dank aussprechen.

Diese Arbeit wäre auch nicht zu Stande gekommen ohne die Kraft und die Motivation, die ich in den folgenden Worten des zeitgenössischen Künstlers Michelangelo Pistoletto gefunden habe:

„Il Terzo Paradiso è il nuovo mito che porta ognuno ad assumere una personale responsabilità in questo frangente epocale. “ (Pistoletto 2003-2012)

2. NLP und CALL

In diesem Kapitel werden die Begriffe der *Verarbeitung der natürlichen Sprache* (NLP) und *Computer-Assisted Language Learning (CALL)* kurz erklärt. Darüber hinaus wird der Zusammenhang zwischen den zwei Bereichen umrissen.

2.1. Der Ursprung der Lehrmaschine

Lehren mithilfe einer Maschine ist keine Erfindung dieser Zeit: Die erste „Lernmaschine“ wurde im Jahr 1809 in den USA entwickelt. H. Chad patentierte eine *Mode of Teaching Reading Machine,* die, wie der Name selbst erklärt, lesen beibringen sollte. Nach nur 100 Jahren wurden diese ursprünglich theoretischen Pläne allmählich in die Tat umgesetzt und erste Prototypen entwickelt.

So träumte im Jahre 1912 der Psychologe E. Thorndike von einem automatisierten Buch:

> „if, by some miracle of mechanical ingenuity, a book could be constructed in such a way that page 2 only became visible for those who have completed page 1 and so on, then much of that which currently requires personal instruction could be performed by the book“ (Thorndike 1913).

Somit geht die Erfindung von automatischen Lehrmaschinen auf den Anfang des 20. Jahrhunderts zurück - in Erwiderung auf die Kritiken an traditionellem Lehren, das nicht immer die Bedürfnisse der Lernenden zufriedenstellte.

Die erste Lehrmaschine wurde 1924 von Sidney Pressey (1927) gebaut: Vier Druckknöpfe entsprachen den möglichen Antworten einer Multiple-Choice Frage und der Lernende konnte erst zur nächsten Aufgabe weitergehen, wenn er eine Frage richtig beantwortet hatte. Die Maschine konnte die Antworten korrigieren und gleichzeitig den Überblick über jede Aktion des Lernenden behalten.

Jedoch wurde Pressey wegen mangelnden Kenntnissen über die Eigenschaften des Lernens stark kritisiert (Skinner 1968). Der Fokus wurde deswegen auf die Lernphänomene und die Entwicklung des programmierten Lernens gelegt.

Skinner bezog sich auf seine eigenen Werke über Behaviorismus und er bestätigte, dass Konditionierung die Lernmechanismen beeinflussen könnte.

Darauf basierend stellte er sich eine wissenschaftliche Art des Lehrens vor, die linear programmiertes Lernen nutzte, das von einer Maschine gelehrt werden sollte.
Nach Skinners Meinung liegen fünf Prinzipien dem wirkungsvollen Lernen zugrunde:

1. Das „active responding“ Prinzip: Das Subjekt muss nicht nur die korrekte Antwort erkennen, sondern auch konstruieren können (die Multiple-Choice-Fragen können Fehler verursachen, welche die Subjekte sonst nie machen würden);
2. Das „small steps“ Prinzip: Das Material wird in mehrere kleine Abschnitte gegliedert, sodass leistungsschwächere Lernende sie sorgfältig befolgen können;
3. Das Prinzip der graduellen Progression;
4. Das „self-pacing“ Prinzip: Die Lernenden sollen ihr eigenes Tempo festlegen;
5. Das Prinzip der korrekten Antwort: Die Lernenden sollen angeleitet werden, da zahlreiche fehlerhafte Antworten entmutigen.

Die Übungen der von Skinner 1953 gebauten Maschine waren auf einer gezackten Rolle, die die Lernenden selbst kurbelten. Laut des Psychologen sollten während einer Lernsitzung nicht mehr als 10% aller Antworten falsch sein.
Doch schnell wurden Kritiken an Skinners Ideen laut und 1959 schlug Norman Crowder ein alternatives System vor (1963). Im Vergleich mit Skinners Maschine unterschied sich seine Maschine in vielen Aspekten - besonders in den Aufgabentypen und wie diese präsentiert wurden.
Crowders Maschine erläuterte das Material zunächst mit Informationen und stellte daraufhin eine Multiple-Choice-Frage. Im Gegensatz zu Skinner sprach Crowder Fehlern eine wichtige Funktion zu. Wenn in seinem System die Antwort des Lernenden richtig war, konnte er mit den Aufgaben weiter machen, andernfalls wurden ihm helfende Aufgaben gestellt, damit er daraufhin die fehlerhafte Übung allein korrigieren und richtig beantworten konnte. Diese Art von Lehrmaschine konnte sich besser auf die individuellen Lerntypen und ihre Lernstrategien einstellen, wodurch diese Erfindung sehr wichtig für die Entwicklung des programmierten Lernens wurde.
De Montmollin beschrieb das programmierte Lernen als „a teaching method which

makes it possible to transmit knowledge without the direct intermediary of a teacher or monitor, whilst taking into consideration the individual characteristics of each pupil“ (1971). Diese Lehrmethode stützte sich auf vier Prinzipien:

1. Das Prinzip der Strukturierung des Themas: Die Themen werden ausgewählt und vorgestellt, um das Verständnis und das Auswendiglernen zu erleichtern;
2. Das Anpassungsprinzip: Das Unterrichten sollte dem Lernenden angepasst werden;
3. Das Prinzip der Stimulierung: Der Lernende muss weder sich langweiligen noch sich demütigen; er muss an dem Unterricht aktiv teilnehmen;
4. Das Prinzip der Kontrolle: Die Erlernung muss dauerhaft kontrolliert werden.

Der Lernprozess mit einer Lernmaschine kann entweder linear (z.B. Skinner) oder verzweigt (z.B. Crowder) programmiert sein.
Das programmierte Lernen eröffnete neue Forschungsmöglichkeiten für die Methoden und Theorien des Lehrens und Lernens. Dies bezeichnet den Anfang von CAI (*Computer-Assisted Instruction*).

2.2. Von CAI zu CALL

Am Anfang der sechziger Jahre entstand das computergestützte Lernen – *Computer Assisted Instruction* - und der Computer automatisierte zunächst den mechanischen Teil der Lernmaschine.
Das beinhaltete in erster Linie sich wiederholende Aufgaben, die auf einer behavioristischen Methode für Lernen und Tutorien innerhalb eines kognitiven Ansatzes beruhten.
Erste Versuche, CAI intelligent zu gestalten, gab es erst in den 1970er und 1980er Jahren. Diese Forschung sollte die vorhandenen Einschränkungen ausgleichen:

1. Dialog mit den Lernenden in natürlicher Sprache;
2. Auswahl des benötigten („needed to be taught“) Lehrmaterials;
3. Vorhersage, Diagnose und Verstehen der Fehler der Lernenden;
4. Verbesserung und Veränderung der Lehrmethoden in Abhängigkeit der Lernenden.

In dieser Zeit entstanden Micro-Welten[1], deren Aufgabe es war, eine Umgebung für die Entwicklung abstrakter Bereiche zu bieten - im Gegensatz zu intelligenten Tutoren, die Computer, basierend auf kognitiven Theorien, in Lehrer verwandeln sollten. Diese neue Welle beinhaltete weiterhin akademische Projekte, die Designer, Programmierer und Sprachlehrer zusammen beschäftigten. Die bekannteste Micro-Welt ist die *Tortoise mini-language* von Seymour Papert (1981): Diese Software benutzte als erste einen Konstruktivistischen Ansatz.

Kooperative Systeme und interaktive Lehrumgebungen kamen ab den 1990er Jahren in Gebrauch. Diese Phase beinhaltete dann auch die Entwicklung eines neues Akronyms: CALL (*Computer-Aided Language Learning*); dieses neue Gebiet konnte in kurzer Zeit eigene Themen und spezifische Werkzeuge festlegen. Es hatte die Welt des Lernens und der Wissenschaft in Aufruhr versetzt und begründete die Entstehung einer Vielzahl von spezifischen Studien, Produkten und Journalen. Zudem versuchten die frühen Forschungen in CALL, die Überlegenheit der Computerbenutzung gegenüber den traditionellen Sprachlehrsystemen zu beweisen.

Mangenot (1997) teilte die CALL-Übungen in fünf Kategorien. Die Bedeutung jeder Kategorie ist abhängig von der Umgebung (DVD/Web), seiner Benutzung (mit einem Lehrer, selbst-lernen, etc.), und dem Alter des Lernenden, etc.

1. Hör- und Leseverständnis: Die schnelle Computerentwicklung ermöglichte den Gebrauch von Audiosequenzen und später auch Videosequenzen in Hör- und Leseverständnistests. Die Aufgaben waren entweder Multiple-Choice-Fragen oder Objekte (z.B. Audiosequenzen, Videosequenzen, Wörter, etc.), die anzuklicken waren. Die Korrektur wurde in Form von „richtig oder falsch“ gegeben; ein schriftlicher oder wörtlicher Kommentar – wie z.B. „bravo“ - waren teilweise auch integriert;

2. Übungen, die Diskurswissen vermitteln sollten: Puzzles, Reihenfolgen- und Paarbildungen; die Korrektur wieder in Form von „richtig oder falsch“;

[1] Der Begriff *MircoWorld* bezeichnet eine „begrenzte Welt“, welche die Semantik eines Computerprogramms für die Vorplanung oder das Verständnis der natürlichen Sprache einschränkt; dieses System benutzt die *Logo Programmiersprache* (ein Dialekt der Programmiersprache *Lisp*). Ziel und Zweck ist es, dass Kinder durch Experimentieren komplexe und abstrakte Ideen entwickeln. Als Lernhilfsmittel ist Logo flexibel und gibt direktes Feedback.

3. Sprachaufnahmen: mündliche Übungen zum Umstellen bzw. Umformulieren von Aussagen: Diese Aufgaben beinhalteten entweder Wiederholungsübungen oder Umformulierungen von Aussagen. Der Lernende konnte sich dabei selbst aufnehmen und danach anhören. Einige Softwares benutzten die Stimme des Lernenden, um ein Feedback zu geben und andere verglichen sein Sonagramm mit dem der Person, die sie imitierten. In beiden Fällen wurde eine Form der Selbst-Korrektur („auto correction“) verwendet;

4. Simulationen: Laut Mangenot gibt es drei Simulationsarten: Der User sollte eine Wahl zwischen zwei verschiedenen Aufgaben treffen; der User musste sich mit der Übung vertraut machen und sie wiederholen (z.B. einen Lebenslauf schreiben); der User musste Bilder verbinden, um eine Reaktion des Systems zu erhalten. Bei diesen Aufgaben wurden keine Korrekturen, sondern Kommentare gegeben;

5. Schriftliche Aufgaben: Aufgaben, die ein freies Schreiben verlangten, waren selten Teil einer Software, da die Korrektur zu komplex war. Probleme traten bereits bei Fragen auf, die nur kurze Antworten verlangten (z.B. der Lückentext), doch bei freiem Schreiben sind diese noch komplexer, da Syntax und Semantik in Betracht gezogen werden müssen. Die Korrekturen in CALL waren entweder klassische Verbesserungen in Form von standardisierten Antworten, mit denen der Lernende seine Fehler vergleichen musste, oder *guidelines* (z.B. „do not forget to say that...“, „be more assertive“, etc.).

CALL Software haben wenige selbst-korrektive Aufgaben bezüglich schriftlicher oder oraler Kompetenzen. Die am weit verbreitetsten Übungen sind die, die für den Computer am einfachsten zu korrigieren sind: Der Lernende wählt eine vorgegebene Antwort aus, z.B. Multiple-Choice, Quiz, Zuordnen, *drag and drop*, Koppelung, Lückentext, etc. Trotz der scheinbaren Variation kommen alle diese Aufgaben aus demselben Paradigma der geschlossenen Frage heraus.
CALL entwickelte sich zu einer Zeit, in der der technologische Fortschritt und die mechanischen Vorstellungen des Lernprozesses verglichen zu heute noch

eingeschränkter waren. Es weist daher einen noch eher beschränkten Zugang zur Sprachdidaktik auf – dies zeigt sich vor allem in den Ideen der linearen Abfolge und des konditionierten Lernens.

Somit besteht die Aufgabe, die ganzen Möglichkeiten von CALL auszuschöpfen, indem die ganze moderne Technologie berücksichtigt wird. Forscher in Informations- und Kommunikationstechnik haben sich bereits mit dieser Herausforderung beschäftigt, aber durch Zusammenarbeit mit Nachbardisziplinen wie *Natural Language Processing* (Verarbeitung der natürlichen Sprache), die auch im Bereich der Informationstechnologie angesiedelt sind, kann noch mehr erreicht werden. Der nachfolgende Abschnitt soll daher die Themenbereiche NLP und CALL zusammenführen.

2.2.1. Computer-Assisted Language Learning

Computer-Assisted Language Learning (CALL) ist das Ergebnis aus einer Summe vieler verschiedener Forschungsfelder – z.B. pädagogische Innovationen -, die sich mit der Computernutzung beim Sprachlernen befassen.

Laut Levy (1997) ist der CALL-Bereich dynamisch und kompliziert zu studieren, da er mit ständigen Technologieveränderungen und der Integration der Technologie in das Lernen einer Sprache kämpfen muss.

CALL umfasst eine umfangreiche und verschiedenartige Reihe von Methoden und Ansätzen zum Sprachlernen und -lehren, die alle den Computer nutzen. Diese Methoden und Ansätze werden in vielen Staaten und für verschiedene Sprachen – bzw. Sprachniveaus - genutzt.

Die Forschung innerhalb von CALL zielt darauf ab, theoretische Grundlagen festzumachen, die eine effiziente Entwicklung und eine wirksame Anwendung von CALL ermöglichen.

Die pädagogischen Innovationen, die Forschung und Entwicklung werden in einer von Eurocall – *European Organization for CALL* (Eurocall, Calico, & IALLT 1999) - und von CALICO – *North American Organization for CALL professionals* (Calico, Eurocall, & IALLT 1999) - adoptierten Richtlinie festgelegt.

Levy (1997:1) beschreibt CALL wie folgt:

„Computer-Assisted Language Learning (CALL) may be defined as 'the search for and the study of applications of the computer in language teaching and learning'. The name is a fairly recent one: the existence of CALL in the academic literature has been recognizable for about the last thirty years. The subject is interdisciplinary in nature and it has evolved out of early efforts to find ways of using the computer for teaching or for instructional purposes across a wide variety of subject areas, with the weight of knowledge and breadth of application in language learning ultimately resulting in a more specialized field of study."

Somit wird CALL als ein interdisziplinäres Gebiet betrachtet. Levy stellt selbst eine Liste von Disziplinen zusammen, die im Laufe der Zeit CALL beeinflusst haben[2].
Es wurde außerdem von unterschiedlichen Lerntheorien beeinflusst. Warschauer (1996) definiert drei CALL-Phasen:

1. Behavioristische Phase, die auf behavioristischen Lehrtheorien basiert ist;
2. Kommunikative Phase, die auf den kommunikativen Ansatz in Lerntheorien gründet;
3. Integrative Phase, die laut Warschauer auf Multimedia und Internet basiert ist[3].

Eine weitere Erläuterung zu CALL liefert Beatty: „a definition of CALL that accommodates its changing nature is any process in which a learner uses a computer and, as a result, improves his or her language" (2003:7). Diese Verbesserung der Sprache betrachtet Beatty von verschiedenen Standpunkten aus:

1. Lerneffizienz: Die Lernenden können schneller und kraftsparender ihre Sprachkenntnisse oder -fähigkeiten erhöhen;
2. Lernwirksamkeit: Die Sprachkenntnisse oder -fähigkeiten bleiben den Lernenden langfristig erhalten; sie assoziieren tief gehender und/oder erzielen besser Lernergebnisse, bei den Dingen, die sie lernen sollen;

[2] Applied Linguistics, Artificial Intelligence, Cognitive Psychology/Science, Computational Linguistics, Curriculum development, Educational psychology, Educational technology, Expert systems, Human-computer interaction, Information processing, Instructional design, Instructional technology, Language data processing, Language teaching methodology, Linguistics, Machine translation, Materials design, Natural language processing, Parsing theory, Programmed instruction/learning, Psycholinguistics, Second language acquisition, Sociolinguistics, Systems theory. (1997:49)

[3] Für weitere Informationen zu diesen drei Phasen siehe Warschauer (1996), und Heift und Schulze (2007).

3. Zugang: Die Lernenden haben direkten Zugang zu den Materialien oder Interaktionen, die ansonsten schwierig zu erreichen wären;

4. Komfort: Die Lernenden können zu den unterschiedlichsten Zeiten an verschiedenen Orten erfolgreich lernen, ohne sich an feste Pläne halten zu müssen;

5. Motivation: Das Lernen bereitet Vergnügen, wodurch sie sich intensiver und fleißiger damit beschäftigen;

6. Institutionelle Effizienz: Sie benötigen weniger Betreuung von einem Lehrer, beziehungsweise generell kostengünstigere Ressourcen.

2.2.2. Natural Language Processing

Natural Language Processing – die Verarbeitung natürlicher Sprache – ist ein Zweig der künstlichen Intelligenz (KI), die versucht, Aspekte der menschlichen Intelligenz auf Maschinen zu übertragen. Mithin behandeln KI-Forscher Wissensdarstellung, -argumentation und -suche und wenden diese Algorithmen und Techniken auf Spiele, Lehrmaschinen, Computervision, Robotik, Expertensysteme und Intelligente Agenten an (Sharples 1989; Finlay, Dix 1996).

Die Verarbeitung der natürlichen Sprache, die sowohl *Natural Language Understanding* als auch *Natural Language Generation* enthält, ist direkt mit formaler Linguistik, Computerlinguistik, als auch mit der Kognitionswissenschaft verbunden.

Nach Jurafsky (2006:2) gilt:

> „by speech and [natural] language processing, we have in mind those computational techniques that process spoken and written human language, as a language. What distinguishes these language processing applications from other data processing systems is their use of knowledge of language. “

Die in NLP bearbeiteten Sprachaspekte sind üblicherweise das Lexikon, die Morphologie und die Syntax. In den letzten Jahren werden aber auch die Perspektiven der Semantik, des Diskurses und der Pragmatik immer bedeutender und werden daher stärker in die Forschung integriert.

In diesem Bereich kann man zwei verschiedene umfassende Anwendungsgebiete der NLP nennen: Einerseits wird NLP verwendet, um die Sprache - Wörter, Sätze, Texte - des Lernenden zu analysieren; das beinhaltet den Ausbau der Tutorensysteme in

ICALL[4] (*Intelligent Computer-Assisted Language Learning*) und die Analyse und die Annotierung der Korpora. Andererseits kann NLP auf die Analyse der Muttersprache angewandt werden: der Zugriff auf muttersprachliche Lesestoffe und Korpora, als auch die Erzeugung der auf muttersprachlichen Materialien basierenden Übungen, Spiele und Texte.

Diese Arbeit wird sich mit dem ersten Punkt befassen. *Intelligent Language Tutoring Systems* (ILTS) benutzen NLP, um den Lernenden persönliche Feedbacks zu geben, während sie Aufgaben machen, z.B. aus Übungsbüchern wie in *E-Tutor* (Heift 2010), *Robo-Sensei* (Nagata 2009) oder *TAGARELA* (Amaral, Meurers 2011).

Die NLP-Analyse kann das Material an den Lernenden persönlich anpassen und das *Student Model*[5] aktualisieren. Die Analyse fokussiert auf die von den Lernenden gemachten Fehler, aber das Feedback kann gleichermaßen korrekte Strukturen oder die Angemessenheit einer Antwort betreffen.

2.3. Wie kann CALL von NLP profitieren?

Es scheint, dass NLP natürlich im Bereich des Sprachlernens angewandt wurde. ICALL, die Kombination aus NLP und CALL, hat drei mögliche Grundsätze:

Erstens, die Fähigkeit der Fehlererkennung in der Eingabe des Lernenden und die Erzeugung eines Feedbacks (Heift 2002; Nagata 1993).

Zweitens soll es sich an das Leistungsniveau des Lernenden anpassen und entsprechende Aufgaben und Materialien erstellen.

Drittens, die Interaktion mit dem Lernenden mittels programmierter Einheiten, die menschliche Gesprächspartner simulieren, und angemessen auf den Input des Lernenden reagieren und antworten können.

In den Achtzigerjahren, besonders zwischen 1985 und 1995, wurde erstmals NLP in der Sprachdidaktik benutzt (Swartz, Yazdani 1992; Chanier 1993; Holland et al. 1995; Jung 2005). Die Nachfrage nach Computern stieg rasch und es wurden folglich die ersten Versuche zur Eingliederung der Informations- und Kommunikationstechnik in den Unterricht gemacht. Eine Reihe von Entwürfen und Prototypen für die

[4]Eine nähere Erläuterung zu ICALL im nachfolgenden Abschnitt 2.3.

[5]Ein *Student Model* ist ein Profil, das das System für jeden Lernenden anlegt, um dort dessen Lernfortschritte zu speichern. Basierend auf dieses *Student Model* werden dann dem Lernenden angepasste Übungen gestellt.

Sprachanalyse und Spracherzeugung wurden entwickelt und diese markieren den Ansatzpunkt für die Schöpfung neuer Sprachdidaktik-Systeme.

Im Allgemeinen waren die ersten Systeme schwerfällig und erreichten sofort die Grenzen ihres Potenzials. Des Weiteren wurde die Problematik des Lernens einer zweiten Sprache - also einer Nicht-Muttersprache – im Gegensatz zum Erstspracherwerb nicht in Betracht gezogen. Dies verhinderte letztlich eine Zusammenarbeit von Computerspezialisten bzw. CALL-Spezialisten und Sprach- oder Lehrexperten.

Ende der 1990er Jahre wurde diese Problematik überwunden und es wurden viele neue fächerübergreifende Arbeiten und Konferenzen in diesem Bereich durchgeführt.

Für das Modell des Informations- und Kommunikationssystems wurden vier neue Komponenten festgelegt, mit denen der Lernende interagieren sollte:

1. Ein Teil, der die grundsätzlichen Sprachkenntnisse (*linguistic knowledge*) regelt;
2. Ein Teil, der den Mensch-Computer-Dialog regelt;
3. Eine Komponente, welche sowohl die Kenntnisse als auch die Fehler der Lernenden feststellt;
4. Eine Komponente, welche die didaktischen Interventionen des Systems regelt.

Durch dieses Modell bzw. diesen Aufbau wurde CALL für Sprach- und Lehrexperten interessanter; zudem erlaubte die rasche Entwicklung des Internets einen weiteren Vorteil: Die Lernenden können auf einfachste Weise auf ein reiches Angebot an Lehrmaterialien zugreifen und problemlos mit Muttersprachlern verschiedener Sprachen verbunden sein.

Die aktuelle Forschung und Arbeit möchte sich aller realistischen Ergebnisse der NLP-Forschung bedienen und sie auf die Ziele von CALL übertragen. Häufig werden für eine bestimmte CALL-Anwendung spezifische Sprachverarbeitungsprogramme entwickelt. In diesem Zusammenhang zeigt NLP seine Relevanz für CALL Systeme: „it is enriched through the input of language teaching specialists on the issues at stake in language, its demands and limitations. “ (Antoniadis et al. 2011:5).

An dieser Stelle sollen drei Projekte kurz skizziert werden, die Beispiele für ICALL-Systeme darstellen.

Das erste Großprojekt war das *PLATO*[6] System, das an der Universität von Illinois gestartet und ausgebaut wurde. Es benutzte einen Ansatz mit programmierten Anweisungen, die dem Lernenden Aufgaben stellten, die seinem Sprachniveau angepasst waren; wenn nötig wurden ein Feedback und zusätzliche unterstützende Übungen gegeben. Die Inhaltsstoffe wurden für zahlreiche Sprachen, vor allem aber für das Französische, entwickelt (Levy 1997).
Der Hauptgrund für die Entwicklung von PLATO war wie folgt: „the system was designed to maintain detailed records of value not only for the teachers and students but also for researchers“ (Hubbard 2009:4).

Ein weiteres Projekt ist das *Athena Language Learning Project*[7] (Massachusetts Institute of Techology), das versuchte einen Zusammenhang zwischen interaktiven Bildplatten und künstlicher Intelligenz zu erstellen, um die Sprachdidaktik bzw. das Sprachlehrsystem umzukrempeln. Obgleich die KI-Elemente nicht vollkommen definiert wurden, erzielte das Projekt einige hochinteressante Ergebnisse, insbesondere *A la recontre de Philippe*, ein Mitbestimmungsdrama in dem die Lernenden eine Rolle spielen und eine Wohnung in Paris suchen (Murray et al. 1989).

E-Tutor[8] ist ein weiteres unkommerzielles ICALL Web-basiertes System für Lernende des Deutschen, in dem NLP und künstliche Intelligenz in CALL zusammengebracht werden. Es war für Studenten gedacht, denen die Hauptteile der deutschen Grammatik beigebracht werden sollten. Das System folgt dem Inhalt und Lexikon eines in Nordamerika gebräuchlichen Lehrbuchs. Seit 1999 wurde es implementiert und weiterentwickelt, sodass es bessere Funktionalitäten und allgemeine Verbesserungen - durch das starke Nutzen des *Student Models* und des Feedbacks - zeigte (Heift, Schulze 2007; Heift 2010).

---

[6] *Program Logic for Automatic Teaching Operations* ist das erste Computer Assisted Instruction System, das im Jahr 1960 entwickelt wurde.

[7] Das Projekt nimmt an einer großen Reihe von Experimenten innerhalb der Softwareproduktion teil. Das MIT arbeitete mit IBM und Digital Equipment Corporation zusammen, um Prototypen für Sprachkurse in Französisch, Deutsch, Spanisch und Englisch als Fremdsprachen zu kreieren.

[8] Entwickelt an der Simon Fraser University, Burnaby, Kanada; für weitere Informationen siehe Heift 1998b, 2001, 2002, 2003, 2004; Heift, Nicholson, 2000a, 2000b.

Ein wichtiger Leitfaden in CALL ist die interdisziplinäre und kooperative Arbeit. Jeder Bereich muss sich auf den für ihn wichtigen bzw. passenden Teil des Begriffs „Computer-Assisted Language Learning“ konzentrieren: Die Didaktik definiert das Wort „Learning“; die Sprachwissenschaft regelt das Wort „Language“ und die Informationstechnologie den Begriff „Computer-Assisted“. Unter der Bezeichnung Computer soll nicht nur der typische Rechner verstanden werden, sondern auch das Netzwerk und viele andere Zusatzgeräte – bspw. PDA, mp3, Handys, interaktives Whiteboard, DVD-Players (Levy, Hubbard 2005). Darüber hinaus müssen die Fächer zusammenarbeiten und zur allgemeinen Lösung beitragen.

2.3.1. Übertragung von NLP-Technologien in CALL

Zahlreiche Technologien wurden von CALL in Anspruch genommen, um linguistische Strukturen darzustellen, um die Sprache zu erklären oder um verschiedene Übungen zu geben und die Fehler zu korrigieren.

1. Konkordanz: Die Konkordanzprogramme sind wohlbekannte und weitverbreitete Sprachbearbeitungsprogramme, die kompetente Sprachlehrer inspiriert haben;

2. Text Alignment: Wenn automatische Textalignierungen auf zweisprachige Texte angewandt werden, bietet der resultierende Text umfangreiche Informationen für fortgeschrittene Sprachlernende;

3. Spracherkennung und -synthese: Die Sprachtechnologie kann die Aussprache einzelner Wörter, generieren; das befreit die Sprachlernenden von der Notwendigkeit das IPA-System (Internationales Phonetisches Alphabet) zu kennen. Die Spracherkennung kann die Aussprache überprüfen, wenn möglich verbessern;

4. Morphologische Verarbeitung: Die Lemmatisierung und die morphologische Generierung werden eingesetzt, um Übungen für Lernende zu erstellen, um die Wörterbuchsuche zu unterstützen (die in flektierten Sprachen ohne

morphologische Analyse unmöglich ist), und um den Zugriff auf Korpora flexibler zu gestalten;

5. Syntaktische Verarbeitung: Die syntaktische Generierung kann neue Aufgaben kreieren und der Parser wird sowohl für die Verdeutlichung der Sprachstrukturen, als auch für die Fehlererkennung und Fehleranalyse in den Übungsaufgaben genutzt;

6. Die Maschinelle Übersetzung schließt den größten Teil der in CALL benutzten Sprachtechnologie mit ein (Levin 1993).

2.3.2. Die Benutzung von Korpora im CALL Bereich

Die Nutzung von Korpora beim Sprachlernen ist ein wichtiges NLP- und IT-Anwendungsgebiet, da eine große Menge von Informationen und eine Reihe von realistischen Sprachsituationen und Beispielen in Korpora gefunden werden können.
Moderne Arbeiten zeigen, wie man aus authentischen Texten bestehende Korpora für die Erzeugung von Übungen benutzen kann (Selva 2002; Antoniadis et al. 2011).
Somit stellen Korpora eine breit gefächerte Quelle mit Beispielen, insbesondere mit linguistischen Phänomenen dar – von Lexikon über Morphologie, bis hin zu Syntax, Phraseologie, Terminologie und Übersetzung (im Falle von mehrsprachigen Korpora).
Texte werden mit Hilfe einfacher Parser – bspw. *Finite State Automata* – getaggt und lemmatisiert, aus denen dann diese Informationen gezogen werden können.
Die Arbeit mit Korpora stellt für Lernende eine nützliche Ressource dar, mit der sie Kookkurrenzen eines gegebenen Ausdrucks finden können, eine Struktur kontrollieren oder eine Bedeutung bestätigen können, etc.
Deville et al. (2004) schlagen vor, diese Funktionsweise als Verständnishilfe zu benutzen.
Der daten-gesteuerte Lernansatz führt zu einem hohen Arbeitsaufwand, Quellen und Systemen (Tribble, Barlow 2001; Granger et al. 2001; Granger 2002), die mit der Ergänzung einfacher NLP-Techniken angereichert sein können. Dieser Lernansatz (Johns 1994) unterstützt den Lernenden, damit er durch Computerapplikationen die Grammatik und den Wortschatz erkunden kann. Ein Beispiel ist das Konkordanz-

Programm, das dem User erlaubt, ein Element - z.B. Wort, Konstituente oder Satz - auszuwählen und das dann in einem Korpus nach anderen Entsprechungen dafür sucht. Wenn Lernkorpora zusammengestellt und analysiert werden, kann man die Korrekturen der Textproduktion des Lernenden automatisieren. In zwei verschiedenen Gebieten können Lernkorpora für die Erkennung und Analyse von Fehlern als wichtig betrachtet werden: Erstens, sie erlauben die Extraktion von passenden *mal-rules*, die auf typischen Fehlern basieren, d.h. Fehlern die in speziellen Korpora häufig vorkommen. Zweitens, das Lernkorpus kann als Maßstab zur Bewertung der Ergebnisse angesehen werden – mittels verschiedener NLP-Techniken.

Ein fehlerhaftes Korpus ist von Nutzen, um mit gewissen Techniken Fehlertypen zu bestimmen, um somit ein effizienteres Fehlererkennungssystem bilden zu können.

Das Internet ermöglicht zudem die gemeinsame Nutzung der Ressourcen und Systeme, die ein entscheidender Faktor für die massive Ausweitung der Korpora im Sprachlehrbereich darstellen. Viele CALL-Systeme benutzen entweder annotierte oder nicht-annotierte Korpora: Die Genauigkeit beim Kompilieren und Taggen neuer Korpora und ihre Relevanz in speziellen Situationen hat auf die Qualität der Systeme starken Einfluss.

Im Gegensatz zur automatischen Analyse der Sprache des Lernenden, wie es in einem ILTS System gemacht wird, bietet die Annotierung eines Lernkorpus ein Verzeichnis der Spracheigenschaften. Dadurch können der L2-Erwerb in Fremdsprachenlehr- und -lernforschung (FSLLF) gründlicher begriffen, und neue förderliche Methoden und Materialien in der Fremdsprachdidaktik (*FLT – Foreign Language Teaching*) entwickelt werden.

2.3.3. NLP und CALL für Fehleranalyse

Wie schon erklärt, können Computer den Lernenden beim Entwickeln einiger Fertigkeiten helfen, bspw. mündlicher Fähigkeiten, Lese- und Schreibfähigkeiten und grundlegender Sprachkenntnisse. Die vorliegende Arbeit behandelt keine Themen in Bezug auf Hörverständnis, mündliche Produktion und Aussprache, aber dennoch soll erwähnt werden, dass es viele Systeme gibt, die auf Hörverständnis und visuelle

Videosequenzen spezialisiert sind. Für das Üben der Aussprache gibt es beispielsweise drei verschiedene Arten von Applikationen: Die Sprachaufzeichnung – um die eigene Aussprache direkt mit der korrekten vergleichen zu können -, Sprachvisualisierung und drittens die Automatische Spracherkennung.
Computerprogramme können auf drei Weisen an der Leseentwicklung mitwirken: Erstens, sie kontrollieren was die Lesenden sehen und wie lange sie es sehen, um die Lesestrategien und Automatisierung zu erkennen; zweitens stellen sie Leseverständnistests, und drittens Fußnoten und andere Hilfsmittel zur Verfügung.
Anfangs fokussierte sich die Arbeit und Forschung zum Schreiben auf zwei Bereiche: Einerseits, das Entwickeln der Textverarbeitungsfähigkeiten des Lernenden und, zweitens, das Nutzen von textbasierten und später graphischen *Organisern*, um den Schreibprozess zu unterstützen. Laut Pennington (2004) werden zum Einsatz von Textverarbeitungsprogrammen keine Forschungen mehr betrieben, weil sie schon positive Auswirkungen auf die Textlänge, Textqualität und -quantität und auf die Qualität der Überprüfung zeigten. Außerdem wird die Textverarbeitung von allen Benutzern verwendet, sie ist sozusagen „normal“ geworden (Bax 2003).
Die ersten Programme konzentrierten sich auf Grammatik oder Entwicklung des Lexikons, da diese Anwendungen verglichen mit anderen einfacher zu programmieren waren.

Da Lernende natürlich immer Fehler machen, fokussiert sich die Forschung im NLP-CALL Sektor auf die Fehlererkennung und -analyse der Lernenden. So zeigten sich ICALL Programme bei der Unterstützung des Grammatiklernens als sehr effizient – wenn sie mit speziellen Strukturen benutzt werden – da hiermit eine Reihe von Fehlertypen vorausgesehen und das Feedback entsprechend geplant werden kann. Laut Allen (1996, 1997) sollen Sprachlehrende nicht jeden kleinen Fehler korrigieren, sondern unter einem pädagogischen Gesichtspunkt sollten zuerst die Fehlertypen und die unterschiedlichen Feedbacks, in enger Zusammenarbeit mit Experten der Fremdsprachdidaktik, festgestellt werden.
Die automatische Analyse kann ebenso wohlgeformte Sprachstrukturen erkennen und positive Feedbacks geben, die z.B. eine richtige syntaktische Struktur oder ein angemessenes Nutzen des Lexikons hervorheben, was natürlich ein positiver Anreiz für die Lernenden darstellt.

Die tiefe Einsicht in den Spracherwerb ist daher ein Kernpunkt für die wirksame Fehleranalyse in NLP.

In muttersprachlichen Textproduktionen sind Fehler gleichfalls vorhanden. Die Notwendigkeit, robuste NLP-Systeme zu entwickeln, die auch in nicht ganz optimalen Situationen funktionieren, war eine treibende Kraft in der Wende von theorie- und regelbasierten NLP-Systemen der 80er und 90er Jahre hin zu den heute dominierenden daten-gesteuerten, statistischen und *machine-learning* Ansätzen.

Jedoch findet sich ein wichtiger Unterschied zwischen dem Ziel der Benutzung von NLP in ILTS verglichen mit anderen Domänen der NLP. Einerseits dient die Robustheit von NLP dazu, Fehler und unerwartete Merkmale der Eingabe zu vertuschen, sodass dennoch ein Ergebnis einer Parser-Analyse oder eines maschinellen-Übersetzungssystems erzeugt werden kann. Andererseits, zielen die ILTS Systeme auf die Erkennung der Spracheigenschaften und deren Abweichung vom geschätzten Target, um dem Lernenden ein passendes Feedback zu geben. Die Fehler sind als bedeutende Faktoren zu betrachten und sie sollten nicht durch die Robustheit des Systems vertuscht werden.

Die Rechtschreibprüfungen – *standard spell* und *grammar checker* (Dickinson 2006) - teilen mit ILTS den Fokus auf Fehlererkennung, aber sie stützen sich auf Annahmen typischer muttersprachlicher Fehler. Rimrott und Heift (2008:73) bemerken, dass „in contrast to most misspellings by native writers, many L2 misspellings are multiple-edit errors and are thus not corrected by a spell checker designed for native writers."

Die Rechtschreibprüfungen wurden als Hilfe für die Verbesserung der Grammatikkompetenz erforscht, jedoch sind auch ihre Einschränkungen zu betrachten und zu verstehen (Burston 2001). Tschichold (1999) weist darauf hin, dass traditionelle Rechtsschreibprüfungen für Sprachlernende nicht besonders nützlich sind, da sie anstatt einer Liste verschiedener Alternativen eher mehr Grundlagen brauchen.

Rechtsschreibprüfungen für Lernende – wie *ESL Assistant* (Gamon et al. 2009) für die englische Sprache – stellen sowohl wichtige Feedbacks, als auch Alternativen für den Lernenden zur Verfügung, um ihm die verschiedenen Alternativen verständlicher zu machen, sodass er sich so für die richtige Variante entscheiden kann.

Der User soll mit Hilfe der Rechtsschreibprüfungen in der Lage sein, einen korrekten und wohlgeformten Text zu schreiben.

Das Wörterbuch ist eine der noch am weitesten verbreiteten Applikationen für den Umgang mit Fehlern und deren Vermeidung durch Feedback: Es ist für die Sprachlernenden sehr wichtig, da es den Umgang mit diskreten Einheiten – Wörtern, Begriffen, Übersetzungen, etc. - mit einbezieht und somit einfach zu programmieren und verwalten ist. CALL ermöglichte das Entwickeln von elektronischen Kommentaren als Hilfsmittel sowohl für die Wortschatzentwicklung, als auch für das Leseverständnis. Forscher haben verschiedene Arten von Kommentaren „geprüft": Kommentare in der L1 (Muttersprache) der Lernenden, als auch in der L2 - die Sprache, die gelernt wird (Yoshii 2006); zusätzlich sowohl Kommentare, die sichtbar und explizit waren als auch welche, die unsichtbar bzw. implizit blieben (DeRidder 2002); und zwischen den verschiedenen Kombinationen von Texten, Bildern und Videos (Chun, Plass 1996; Al-Seghayer 2001), um so die effektivste und beste Form des Feedbacks zu bestimmen. Nach Chuns Studie (2006) können digitale Kommentare – sowohl in L1 als auch in L2 - hilfreich sein; generell werden das Anschauungsmaterial und die Texte gemeinsam betrachtet vorteilhafter und nützlicher, als jedes für sich alleine. So stellen Tanenhaus et al. (1995:1633) bereits fest:

> „Relevant visual context has effects on the rapid mental processes that accompany spoken language comprehension; visual context influenced spoken word recognition and mediated syntactic processing, even during the earliest moments of language".

Die Methoden für die Fehlerdiagnose gliedern sich in zwei allgemeine Gruppen: erstens, die traditionellen Entwicklungen zu diesem Thema beziehen sich auf *language-licensing* Ansätze, welche die ganze Antwort des Lernenden analysieren; zweitens steigt die Zahl von *pattern-matching* Ansätzen, die auf besondere Fehlermuster (*error patterns*) und -typen (z.B. Fehler in Präposition und Determinator) abzielen und die Antwort des Lernenden nur auf diese Muster und Typen überprüfen und den Rest übergehen. Die letzten zwei Begriffe werden ausführlich in Meurers (2012) beschrieben.

3. Vorstellung der Parser

Dieses Kapitel beschreibt die Wichtigkeit von Parsern im Bereich des *Computer Assisted Learning Language*. Dazu werden zwei Parser dargestellt: Zuerst *ParZu*, ein in Zürich entwickelter Parser; danach *GI-Tutor* (GermanItalian Tutor) - ein in Venedig entwickelter RTN/ATN Parser.

Der Begriff „Parsing“ bezieht sich auf die automatische Analyse einer Äußerung, dessen Ziel es ist, die wahrscheinlichste syntaktische Struktur dieser Äußerung zu bestimmen (Nederhof, Satta 2010:105). Der Parser muss eine formale Grammatik (vgl. Abschnitt 3.1.1.) auf den Satz applizieren, um die Grammatikalität des gegebenen Satzes zu überprüfen.

Ein auf einer beliebigen Sprachtheorie beruhender Parser, der auf die Realisierung der Analyse zielt, muss Mehrdeutigkeiten und Ambiguitäten der Sprache berücksichtigen, da sie ein entscheidender Faktor für das vollwertige Verständnis der Satzbedeutung sind. Somit muss diese Theorie drei verschiedene Erfordernisse erfüllen: die psycholinguistische Glaubwürdigkeit, die computationelle Effizienz in der Realisierung, und die Beachtung der Bedingungen und der grammatikalischen Regeln (Delmonte 2008:94). Um eine nützliche Analyse zu bekommen, muss der Parser darüber hinaus ein Grammatikalitäts- bzw. Akzeptabilitätsmaß benutzen, falls die grammatikalischen Regeln einmal verletzt werden.

3.1. Parser in CALL

Laut Levy (1997) leiteten sich die mangelnden Fortschritte in einigen Bereichen von CALL von unzureichender Nutzung der NLP-Techniken und dem Fehlen von *linguistic modeling* ab - wie Kohn (1994) schon ausdrücklich erwähnt hatte.

Zugegebenermaßen schließen nur wenige CALL-Applikationen die Technologien der natürlichen Sprachverarbeitung mit ein. Grund ist hierfür aber auch, dass die Techniken des Parsings und die Kenntnisse über die Linguistik erst später ein Niveau erreicht haben, das die Entwicklung eines voll funktionstüchtigen Parsers erlaubte (Matthews 1993).

Weiterhin sind das Entwickeln eines Parsers mit einer computationellen Grammatik und dessen Integration in eine CALL-Einheit noch eher komplizierte, beschwerliche

und zeitaufwendige Aufgaben. Die Einbindung einiger Sprachtheorien (z.B. aus der Phonologie und Morphologie) in die Sprachtechnologie (*Human Language Technology – HLT*) und deren Umsetzung in CALL-Software wurden ermöglicht.
Gegner des parserbasierten Ansatzes in CALL argumentieren, dass die KI-Architektur noch keine dem menschlichen Kommunikationssystem ähnliche Struktur erstellt habe; deshalb sei sie nicht in der Lage, qualitative Fortschritte in der Gestaltung der CALL-Programme einzuleiten (Salaberry 1996). Salaberry besteht darauf, dass „the most important reason for this failure [of ICALL] is that NLP programs—which underlie the development of ICALL—cannot account for the full complexity of natural human languages" (1996:11). Jedoch bedeutet dies nicht, dass wichtige und interessante Sprachaspekte nicht in einer formalen Theorie erfasst und in eine CALL-Applikation eingeführt werden können.
Zur Verteidigung des Zusammenhangs „Sprachtechnologie und CALL" listet Nerbonne (2003) die in CALL angewandten NLP-Technologien auf (vgl. Abschnitt 2.3.), die das Lernen erleichtern können, und er schlägt intensiveren Austausch vor, um die Missverständnisse zwischen CALL und NLP und zwischen Computerlinguistik und CALL zu lösen. Die Parser der natürlichen Sprache nehmen die schriftliche Sprache als Input und erzeugen daraus eine formale Darstellung der syntaktischen – und manchmal semantischen – Struktur des Inputs.
Nachfolgend wird ein Überblick über Parser gegeben, die für die Fehlerdiagnose verwendet werden - im Zusammenhang mit verschiedenen grammatikalischen Formalismen.

Viele Parser aus dem Bereich des Sprachunterrichts enthalten Komponenten, die Fehler voraussehen und suchen können - im Falle, dass die grammatikalischen Regeln keinen Erfolg erzielen. Ein gebräuchliches Beispiel sind Regeln, die gewollt fehlerhafte Konstruktionen erlauben (*buggy rules*). Dadurch kann der Parser auf diese Weise ungrammatische Sätze verarbeiten und analysieren. Forscher analysierten hierzu fehlerhafte Lernerkorpora, um die *buggy rules* für einen Parser zu bestimmen. Dabei werden statistische Methoden für die Fehlererkennung und -diagnose benutzt.
Die Rolle des Parsers im CALL-Bereich wurde in vergangenen Jahren einer Überprüfung unterzogen (z.B. Matthews 1992b, 1992c; Holland, Maisano, Alderks, Martin 1993; Nagata 1996). Holland et al. (1993:28) behandelten die Möglichkeiten

und die Einschränkungen, die ein parserbasiertes Sprachlehrprogramm haben kann. Nach einem Vergleich von parserbasiertem CALL mit konventionellem CALL folgerten sie, dass parserbasierte CALL-Applikationen den Lernenden mehr Freiraum lassen und hier eine größere Menge an Sätzen von den Lernenden produziert werden. CALL gestattet in diesem Fall die Beurteilung der Schreib- und Ausdrucksfähigkeiten, wobei nicht nur einzelne Wörter oder kurze Strukturen – wie in konventionellem CALL – sondern auch ganze Sätze, und somit der Satzbau, analysiert werden (Holland et al. 1993:31).

Gleichzeitig erlegt das Parsing gewisse Einschränkungen auf. Wie zuvor erwähnt, konzentrieren sich Parser vor allem auf die Syntax des Inputs, d.h. „ICALL may actually subvert a principal goal of language pedagogy, that of communicating meanings rather than producing the right forms“ (Holland et al. 1993:32). Wenn aber andererseits neue Methoden für das Unterrichten von Grammatik umgesetzt werden, können diese Mängel durch einen Fokus auf wichtige kommunikative Fähigkeiten, behoben bzw. verbessert werden.

Andere Autoren (u.a. Loritz 1992) sehen die Bedeutung des Parsers in CALL als positiv und betonen stark die zusätzliche Hilfe für die Lernenden: Die Sprachlehrparser können einen bedeutenden Beitrag zu effizienten Studien über die Sprache leisten; sie müssen deswegen auch *in-depth parser* integrieren, die trotz der Ambiguitäten – die typische Eigenschaft der fehlerhaften Sprache der Lernenden (1992:19) – eine Analyse erlauben.

Die Wirksamkeit der syntaktischen Parser beim Sprachlehren wurde auch in der Praxis untersucht. So bewertet Juozulynas (1994) die mögliche Nützlichkeit der syntaktischen Parser in der Fehlerdiagnose: Nach der Fehleranalyse eines Korpus mit 400 Seiten Umfang, das von amerikanischen Studenten auf Deutsch geschriebene Aufsätze enthält, zeigt er, dass Syntax das problematischste Feld ist – gefolgt von der Morphologie. Laut seiner Studie stammen 53% aller Fehler aus diesen beiden Teilbereichen der Sprache. Dieser Beitrag zur Fehleranalyse des syntaktischen Parsers zeigt weiterhin, dass 80% der Fehler nicht semantischen Ursprungs sind, und somit durch einen syntaktischen Parser eventuell erkennbar gemacht werden können (1994:5). Juozulynas passt eine Fehlertaxonomie[9] an, die aus vier Kategorien besteht:

[9] Diese Taxonomie wurde anfangs von Hendrickson (1979) entwickelt.

Syntax, Morphologie, Orthographie und Lexikon. Weiterhin spricht er für die Teilung der Orthographie in Rechtschreibung und Interpunktion und rechtfertigt dies im Zusammenhang mit syntaktischem Parsing: Teile der Zeichensetzung können durch die syntaktischen Klammern beschrieben werden, wodurch die Interpunktionsfehler mit dem syntaktischen Parser bearbeitet werden können. Laut Juozulynas machen lexikalische Fehler und Rechtschreibfehler einen nur kleinen Teil der Fehlerarten aus. Von ihnen kann ein Teil durch Zugriff auf ein Lexikon bzw. Wörterbuch identifiziert werden. Falls jedoch Wörter im falschen Kontext benutzt werden, so wird der Parser diese ohne spezifische Fehlerregeln im System nicht erkennen können.

Ein Parser wird häufig in Zusammenarbeit mit einem *Spell Checker* angewandt, wodurch ein Großteil der orthographischen Fehler beseitigt werden kann, bevor der Parser mit der eigentlichen Analyse beginnt. Da die Lernenden generell wissen, was sie inhaltlich mitteilen möchten, können viele semantische Fehler größtenteils auch vermieden werden. Eine parserbasierte CALL-Applikation kann bei der Erkennung morphologischer und syntaktischer Fehler eine wichtige Rolle spielen, da sie einen bedeutsamen Anteil der Fehler innerhalb frei formulierter Texte darstellen. Jedoch haben diese Applikationen einige Einschränkungen: Laut Holland et. al (1993:33) sind Parser „not foolproof“, weil sie die Syntax der aktuellen Umgangssprache nicht sorgfältig analysieren können. Diese Problematik muss während der Gestaltung, Umsetzung und Integration einer parserbasierten CALL-Software berücksichtigt werden.

Dennoch ist es für einen Parser nicht notwendig, perfekt zu funktionieren; aber der Aufbau eines ICALL-Parsers muss die potentielle Wahrnehmung der Lernenden mit in Betracht ziehen, d.h. beispielsweise, ob die Lernenden Kenntnisse über die Einschränkungen des Parsers haben.

Die Kosten für das Entwickeln von NLP-Systemen sind eine zusätzliche Einschränkung für ICALL (Holland et al. 1993). Im Vergleich zu konventionellem CALL, benötigt die Erstellung von ICALL-Applikationen die Mitarbeit von Computerlinguisten, erfahrenen Programmierern und die notwendigen Ressourcen für das Bauen und Prüfen einer Grammatik.

Im Laufe der Jahre sammelte sich ausreichend Expertise im Bereich der Sprachtechnologie an, die von einer steigenden Menge an Computerprogrammen

genutzt wird. Holland et al. (1993) versuchen eine Antwort auf die Frage „What are parsers good for?“ zu geben und sie berufen sich hierbei auf das parserbasierte CALL-Programm *BRIDGE*: Dieses Programm wurde zum Deutsch lernen für die amerikanische Wehrmacht ausgelegt, nachdem es zuvor an einer kleinen Gruppe von fortgeschrittenen Deutschlernenden getestet wurde.
Holland et al. stellen die folgenden Punkte vor:

1. ICALL scheint nützlich für form-fokussierte Anweisungen zu sein und gibt den Lernenden die Gelegenheit, ihre Fehler alleine zu bearbeiten und ihre Leistung und ihr Sprachbewusstsein zu verbessern;

2. ICALL ist für Lernende mit bestimmten Eigenschaften nützlich, da diese nachfolgend gelisteten Eigenschaften verzeichnet werden, die den Nützlichkeitsgrad von ICALL beeinflussen können: a. Vorkenntnisse; b. analytische Orientierung; c. Toleranz der Mehrdeutigkeit; d. Selbstvertrauen als Lernende;

3. ICALL ist für die Forschung geeignet, weil der Parser automatisch Antworten analysiert und Fehler erkennt, klassifiziert und speichert; dies erleichtert die Bewertung der grammatikalischen Kompetenz des Lernenden;

4. ICALL hat eine wichtige Rolle in der kommunikativen Anwendung; Holland et al. (1993) führen an, dass parserbasierte CALL-Systeme in graphische Micro-Welten eingebunden werden sollten, da sie helfen können, die grundsätzliche Semantik zu erfassen.

Diese Schlussfolgerungen wurden durch die Prüfung parserbasierter CALL-Applikationen bestätigt. Laut Nagata (1996) werden relevante Ergebnisse nur durch CALL-Programme erzielt, die das Potenzial des Computers vollständig nutzen - d.h. Programme, die ein notwendiges und angemessenes Feedback geben. Sie zeigt, dass angesichts der gleichen Grammatik und Übungen, das Feedback eines intelligenten Computers für die Entwicklung grammatikalischer Fähigkeiten effizienter ist, als die

Antworten eines Übungsbuches[10].

3.1.1. Formale Grammatik

Das Wissen eines parserbasierten Systems findet nicht nur in den Algorithmen, sondern auch in der Darstellung der Sprachphänomene, bzw. der Rechtschreibung, Grammatik, Bedeutung und Textstruktur der Sprache einen Platz.

Die Gestaltung der Grammatik (d.h. Morphologie und Syntax) ist von äußerster Wichtigkeit. Die Haupttypen der Grammatikformalismen, die in CALL-Systemen benutzt werden, sind: *Head-Driven Phrase Structur Grammar* (Pollard, Sag 1987, 1994; Sag, Wasow, Bender 2003); *Lexical Functional Grammar* (Bresnan 1982); *Government and Binding* (Chomsky 1981); *Principle and Parameters Theory* (Chomsky 1986); *Logic Grammars* (z.B. *Metamorphosis Grammar* von Colmarauer 1978); *Definite Clause Grammar* (Pereira, Warren 1980).

Diese Formalismen werden als *feature-based* (eigenschaftsbasierte) oder *rule-based* (regelbasierte) Grammatikformalismen eingestuft. In der ersten Kategorie nach Chomskys Hierarchie (z.B. HPSG) werden die Sprachinformationen als Strukturen der Eigenschaften formell dargestellt: Diese Grammatiken eignen sich einen lexikalischen Ansatz an; die syntaktischen Informationen werden in allen lexikalischen Einträge beschrieben, wodurch sie nur wenige generalisierte syntaktische Regeln benötigen, die die zulässigen Kombinationsmöglichkeiten von Wörtern und Sätzen in großen Einheiten darstellen. Dagegen ist in der zweiten Kategorie (z.B. DCG) der Großteil der syntaktischen Informationen in syntaktischen Regeln dargestellt, wobei allerdings die lexikalischen Einheiten nur gering spezifiziert werden.

Dies wirft spontan die folgende kritische Frage auf: Wie kann eine formale Grammatik der natürlichen Sprache in ein Computerprogramm umgesetzt werden? Die Antwort liegt dabei im Zusammenhang mit dem mathematischen Charakter des Computers: einige Sprachaspekte und -eigenschaften müssen durch Berechnungsmethoden erfasst werden. Die folgende formale Grammatik kann sich so klar abzeichnen:

G(VN, VT, R, S)

[10]Bezieht sich auf das Japanische; vgl. Nihongo-CALI Programm in Nagata (1996).

Die Domäne dieser Funktion – die Reihe der ganzen Eingangswerte - beinhaltet die Wörter der betrachteten Sprache. Der Wertebereich – die Reihe der möglichen Ausgangswerte – bezieht sich auf die Bäume, welche die Sätze der Sprache darstellen. Die Grammatik G wird als Funktion mit vier Argumenten, die wiederum aus je einer begrenzten Anzahl von Elementen bestehen, beschrieben: VN ist die Reihe der non-terminalen Symbole der Sprache - wie zum Beispiel eine Nominalphrase (NP) oder Verbalphrase (VP); VT ist die Menge der terminalen Symbole, d.h. der Wörter; R steht für die Regeln, die die Zusammenstellung der non-terminalen mit den terminalen Symbolen beschreiben; S ist der Anfangspunkt.
Verschiedene Formalismen wurden für die Beschreibung natürlicher Sprachen durch eine formale Grammatik entwickelt. Matthews (1993) verzeichnet acht Hauptgrammatikformalismen, die in CALL angewandt werden: 1. *Various Augmented Phrase Structur;* 2. *Augmented Transition Networks;* 3. *Lexical Function Grammar;* 4. *Systemic Grammar; 5. Tree Adjoining Grammar;* 6. *Incremental Procedural Grammar;* 7. *Word Grammar;* 8. *Preference Semantics.*

Viele andere Formalismen werden nicht erwähnt, wie zum Beispiel CG (*Constraints Grammar*), HPSG (*Head-Driven Phrase Structure Grammar*) oder PPT (*Principles and Parameters Theory*). Jedoch betonte Matthews 1992 die Wichtigkeit der lexikalischen Informationen in CALL bezüglich des Fremdsprachenlehrens und der Fremdsprachlernforschung.

3.1.2. Parsen schlecht geformten Inputs

Anhand einer Sprache L, die eine Grammatik G und einen Wortschatz V besitzt, wird ein Satz S als eine Liste aus V ausgewählter Wörter betrachtet. Da es in G die Regeln (auch Bedingungen oder *Constraints* genannt) für die Zusammenstellung gibt, werden bestimmte Kombinationen der Wörter als ungrammatikalisch verboten. Die Menge möglicher Ketten, die durch das Nutzen von V zusammengebaut werden, heißen die „Schließung von V“ (*Closure of V*) oder V*: Ohne Bedingungen wäre die Anzahl möglicher Ketten unbegrenzt, da jede Vokabel in V unendlich oft wiederholt werden könnte, um eine Kette aufzubauen. Dementsprechend beinhaltet L nur die Kette in V*,

welche die grammatikalischen Regeln befriedigt.

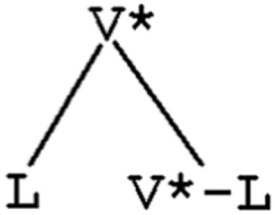

Abb. 1: ***Closure of V and L*** **(Heift, Schulze 2007).**

Die Elemente in V*, die nicht durch die Regeln in G beschrieben werden können, da sie einen oder mehrere Fehler enthalten, finden sich in der Menge V*-L. Angenommen, dass die Anzahl der Sätze einer Sprache grenzenlos ist und jeder Satz eventuell fehlerhaft sein kann, ist die Menge der fehlerhaften Sätze (V*-L) auch unbegrenzt.

Trotz der unterschiedlichen Zwecke behandeln sowohl NLP- als auch ICALL-Systeme das Parsing von schlecht-geformtem Input. Hierbei zielen NLP-Applikationen auf das Parsen und die Analyse eines Satzes – trotz der Fehler –, während ICALL-Systemen auf die Sprachkenntnisse des Lernenden fokussieren - anstatt der Analyse der wohlgeformten Eingabe.

Um ein Feedback zu erzeugen, müssen diese Systeme den Input des Lernenden analysieren und die Abweichung dieses Inputs von den Kenntnissen der Experten bestimmen.

Fehler sind typisch innerhalb ICALL, da sich die Kenntnisse von Lernenden potentiell qualitativ und quantitativ von den Expertenkenntnissen unterscheiden; beispielsweise begehen Sprachlernende Fehler, die durch die Muttersprache beeinflusst sind.

Weischedel und Sondheimer (1983) unterteilen die Fehler in zwei Kategorien: Absolute Fehler beziehen sich auf die Fehler in der Sprachausgabe, z.B. in Genus und Numerus oder in der Satzgliedstellung. Die relativen Fehler bezeichnen grammatikalische Strukturen, die zwar korrekt sind, sich aber jenseits des Zwecks der Grammatik befinden. Die ICALL-Systeme zielen auf die Erzeugung eines Feedbacks für den Lernenden und müssen somit die Fehler des Lernenden analysieren, anstatt sie zu übergehen. Die Fehleranalyse erklärt darauf die Ursache eines Fehlers und erlaubt den Lernenden, die richtige grammatikalische Sprachstruktur zu lernen. Das Übergehen von Fehlern würde diese Absicht eines parserbasierten CALL-Systems

verhindern.

Patten (1992:29) argumentiert: „Parsing is the process by which grammatical strings of words are assigned syntactic structure“. Jedes Wort eines Satzes wird zuerst als Einzelnes analysiert, um seine morphologische Struktur zu bestimmen. Die mit morpho-syntaktischen Eigenschaften markierten Wörter werden danach zu Konstituenten kombiniert - gemäß der grammatikalischen Regeln. Daraufhin wird nach und nach jede Konstituente analysiert und zu größeren Einheiten verknüpft, bis letztlich der ganze Satz geparst ist. Als Ergebnis dieses Prozesses werden entsprechend der Länge und Komplexität eines Satzes eine Vielzahl möglicher Strukturbäume anhand der Konstituentenkombinationen erstellt, bevor der Parser zur finalen Analyse kommt.

Aufgrund der Grammatik und des Lexikons weist ein erfolgreicher Parser auf die wohl-geformten Sätze hin - und, umgekehrt, ein erfolgloser Parser auf die schlecht-geformten Sätze. Es ist wichtig, dass ein Parser die schlecht-geformten Sprachstrukturen behandeln und eine bedeutsame Strukturanalyse erzeugen kann, da sich die Fehler in einem muttersprachlichen Text befinden können: Die Robustheit eines Parsers erlaubt diese Fälle.
Die Parser müssen im CALL Bereich mithin zwei widersprüchliche Zwecke erfüllen: Sie müssen sowohl die wohl-geformten Sprachstrukturen richtig parsen, als auch den Grammatikregeln entsprechend die schlecht-geformten Sätze (Dini, Malnati 1993). Die Regeln für das Parsing der schlecht-geformten Sätze spiegeln die Interimssprache (*Lernersprache*[11]) eines Lernenden wider (Selinker 1974, 1992; Corder 1981). Zahlreiche Ansätze wurden für die Behandlung fehlerhaften Inputs entwickelt, wobei viele dieser Ansätze eine fehlerhafte Grammatik nutzen, welche die typischen Fehler eines Lernenden in einem getrennten Regelsystem erfasst. Die entstehenden Feedbacks können sehr genau sein und die Fehleranalyse ist allgemein verlässlich, weil jeder Fehler an eine bestimmte Regel angehängt wird. Jeder Fehler muss jedoch vorausgesehen werden, beziehungsweise jeder Fehler benötigt eine passende Regel.

11 Interimssprache (Intersprache oder Lernersprache) ist die Zwischenstufe der Sprachkönnens während des Spracherwerbs. Diese Intersprache ist sowohl durch Regeln der Muttersprache und der Zielsprache als auch durch systematische universale Prinzipien geprägt.

Dini und Malnati (1993) listen vier Ansätze für die Behandlung schlecht-geformten Inputs auf[12]:

1. Der *rule-based*-Ansatz stützt sich auf zwei Mengen von Regeln: Eine für die grammatikalische und eine für die ungrammatikalische Eingabe; der Programmierer muss die typischen Fehler vorhersagen, aber wegen der Weite und Unvorhersehbarkeit vieler Fehler kann der schlecht-geformte Input nur stellenweise vorausgesehen werden (Yazdani, Uren 1988). Dazu mangelt die Voraussage der Fehler an Allgemeinheit, da die Systeme die Fehlersuche auf der Muttersprache (L1) basieren.

2. Der *metarule-based*-Ansatz wendet eine Reihe wohlgeformter Regeln an, und falls das System keine benutzen kann, ruft es einen Algorithmus auf, der durch die Anwendung sogenannter *metarules* einige Bedingungen lockert und die Verletzung speichert. Jedoch ist es schwierig, die richtige Entsprechung einer fehlerhaften Struktur festzulegen, vor allem in Bezug auf Kongruenzfehler.

3. Der *preference-based*-Ansatz umfasst eine übergenerierende Grammatik und eine Menge von Präferenzregeln, die eine linguistische Redundanz darstellen.

4. Der *constraint-based*-Ansatz beruht auf den folgenden Annahmen: a. Jeder Baum wird durch einen Fehlerindex markiert; b. Die Verletzung einer Bedingung vergrößert den Fehlerindex des generierten Baums; c. Die Objekte mit dem kleinsten Index werden gewählt (Dini, Malnati 1993). Somit ist die Darstellung des Satzes, die mehrfache Bedingungen befriedigt, die glaubwürdigste.

Viele Autoren haben durch diese Ansätze versucht, eine geeignete Methode zum Parsen schlecht-geformter Eingabe zu analysieren, wobei die Parser sowohl gut als auch schlecht abschnitten (vgl. Heinecke, Kunze, Menzel, Schröder 1998; Menzel 2004; Foth, Menzel, Schröder 2005).

[12]Für weitere Informationen zu dieser vier Ansätze siehe Heift und Schulze (2007).

Diese Methoden erlauben die Erkennung von Fehlern, d.h. Verletzungen der grammatikalischen Regeln, die durch die Programmiersprachen und Grammatikformalismen festgelegt werden. Die ICALL-Systeme werden sowohl in einer prozeduralen Programmiersprache als auch in einer deklarativen Programmiersprache umgesetzt. Die erste Typologie erklärt die Schritte, die der Computer chronologisch anwenden muss. Dagegen enthält der zweite Typ – bzw. *Prolog*[13] – bestimmte Regeln, die der Rechner benutzt, damit er den Kopf der Regel verbinden kann. Beispielsweise:

```
job(susan, department_chair).
job(janet, department_secretary).
who_is(Name,X) :- job(Name,X).
```

Dieses kleine Programm richtet sich auf die Bestimmung von `who_is(janet,X)` und es verknüpft zuerst das Prädikat `who_is`, danach die Variable *`Name`* mit dem Wert `janet` und letztlich den Satz, der aussagt, dass Janet die Sekretärin ist.

Die Programmiersprachen bestimmen die Wahl des Grammatikformalismus, der wiederum festlegt, welche Erkennungsmethode anzuwenden ist, und wie die Fehler vom System wahrgenommen werden (Sanders, Sanders 1989; Matthews 1993, 1994). Jedoch ist die Anwendung von *Prolog* (oder *LISP*) für die Verarbeitung natürlicher Sprachen nicht problemlos, da diese Programme zu spezialisiert und rekursiv sind (Johnston 1988). Laut Handke (1989) wird die Problematik von *Prolog* (oder *LISP*) mit der Implementierung zusätzlicher Hilfsmittel (z.B. *Prolog-based Attribute Logic Engine*) gut gelöst.

3.1.3. Zwei Parsing-Methoden: Konstituenz- vs. Dependenz-Parser

Linguistische Grundlage des Parsers ist die Parsing-Strategie, beziehungsweise die

---

13 Prolog ist eine Programmiersprache (Implementationssprache), welche die Prinzipien des logischen Programmierens realisiert, wie der Name selbst erklärt: *Programming in Logic*. Dieser Aspekt basiert auf der allgemeineren Ansicht, „dass Problemlösen auf das Beweisen der Erreichbarkeit von Zielzuständen aufgefasst wird“ (Bussmann 2008:557). Seit PROLOG zu Beginn der 70er Jahre entwickelt wurde, spielte es seit Mitte der 80er Jahre eine immer bedeutendere Rolle in der Computerlinguistik, vor allem in der Unifikationsgrammatik und der Definite-Clause-Grammatik.

Anwendung der Regeln, die entweder *top-down* (vom Satzknoten zu den terminalen Symbolen) oder *bottom-up* (von den terminalen Symbolen zum Satzknoten) erfolgen kann. Andererseits kann der Parser eine Regelhypothese so lange betreiben, bis er nicht mehr fortfahren kann (*depth-first*), oder an jeder Stelle anfangs alle Alternativen durchgehen (*breadth-first*) - jedoch hat jede Strategie, beziehungsweise Kombination von Teilstrategien, ihre Vor- und Nachteile. Sehr unterschiedlich sind auch die Konzeptstrukturen, die von Grammatikformalismen (vgl. Paragraph 3.1.1.) abhängen.

In der natürlichen Sprache gibt es zwei bedeutende Beschreibungsarten der Satzstruktur: Einmal wird der Satz in Konstituenten (Phrasen) unterteilt, die weiter in kleinere Konstituenten geteilt werden können; oder zweitens, werden die Kanten zwischen einzelnen Wörtern gezeichnet. Während die erste Beschreibungsart den sogenannten Konstituentengrammatiken entspricht, zählt die Zweite zu den Dependenzgrammatiken. Beide Typen sollen hier grundsätzlich behandelt werden. Die Konstituentenphrasengrammatik wurde schon von den alten Stoikern erfunden (Mates 1961) und findet seitdem auch den Weg in die formale Logik und die Linguistik - wie L. Bloomfield, R. Wells, Z. Haris und N. Chomsky. Sie liegt auch der Sprachtheorie, die von Computerfachleuten studiert wird, zugrunde. Dazu erläutert Mel'čuk (1988:3-4):

> „ [P]hrase structure representation in syntax was strongly promoted by the Structuralist school during the thirties, forties and fifties (...). It became the only syntactic representation ever seriously discussed in the work of Noam Chomsky and the Transformational-Generative School he founded in the late fifties. As a result of the triumphal offensive of the transformational-generative approach throughout the world, phrase-structure syntax forced dependency syntax into relative obscurity. "

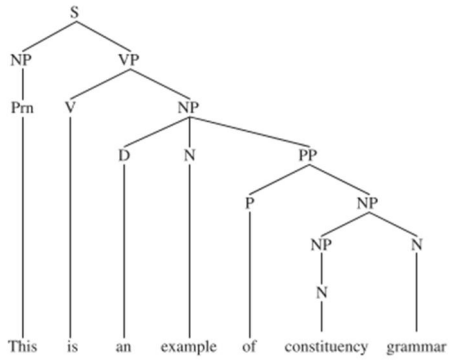

Abb. 2: Ein Konstituenz-Baum.

Auch die Dependenzgrammatik wurde nicht nur einmal formuliert. Jeder Grammatiker zieht den Begriff der Wort-zu-Wort-Verbindung in Betracht, der bspw. die Übereinstimmung, oder die Zuordnung von Kasus erklären möchte. Das Konzept der Dependenz findet sich in altüberlieferten Grammatiken, wie zum Beispiel Latein, Arabisch, Sanskrit. Die Computerimplementierungen dieser Grammatik sind in den letzten 40 Jahren hochinteressant geworden (Hays, Ziehe 1960; Hays 1964; Hays 1966; Fraser 1993; Fraser 1994; Sgall 1994), wurden jedoch nur wenig systematisch erforscht - hauptsächlich wegen des verbreiteten Irrglauben, wonach alle Dependenz-Parser eine Variante des Konstituenz-Parsers seien.

Im Gegensatz zu den Phrasenstruktur-Parsern besteht die syntaktische Dependenzdarstellung aus lexikalischen Einheiten – den effektiven Wörter -, die durch binäre unsymmetrische Dependenzen miteinander in Verbindung gebracht werden: Dies weist eine Ähnlichkeit mit dem Parsingprozess der menschlichen Denkweisen auf (Abney 1989)[14].

Sobald zwei Wörter durch eine Kante miteinander verbunden werden, ist eines der Kopf und das andere der Dependent. Generell ist das abhängige Element Modifikator, Objekt oder Ergänzung, während der Kopf das Verhalten und die Eigenschaften des Paares bestimmt. Der Dependent erfordert die Anwesenheit eines Kopfs, wobei der Kopf abhängige Elemente haben kann, aber nicht benötigt. Die Dependenzstruktur ist ein Baum (DAG – *directed acyclic graph*), dessen Hauptverb die Wurzel (Kopf) ist. Ein Dependent, der seinem Kopf vorangeht, heißt *predependent*; wenn er dem Kopf folgt, heißt er hingegen *postdependent*. Ein Wort ist unabhängig, wenn es kopflos und ohne Dependenzen ist.

Bereits Tesnière (1959) lieferte eine detaillierte Schilderung der Dependenzgrammatik:

> «La phrase est un ensemble organisé dont les éléments constituants sont les mots. Tout mot qui fait partie d'une phrase cesse par lui-même d'être isolé comme dans le dictionnaire. Entre lui et ses voisins, l'esprit aperçoit des

[14] Der Parser im menschlichen Verstand arbeitet wie ein Dependenz-Parser. Angenommen ein Verb kann ein direktes Objekt oder keins haben – z. B. *essen* -, dann muss ein *top-down* Konstituenz-Parser von vornherein wählen, ob er das Objekt erwarten muss, und falls er die falsche Auswahl getroffen hat, muss er einen Rückzieher machen (*backtrack*). Ein *bottom-up* Konstituenz-Parser kann keine Verbphrase bauen, solange er nicht alle notwendigen Wörter gefunden hat. Jedoch wird eine Verbphrase erst dann deutlich verstanden, wenn sie komplett gelesen wird (Abney 1989).

connexions, dont l'ensemble forme la charpente de la phrase. Les connexions structurales établissent entre les mots des rapports de dépendance. Chaque connexion unit en principe un terme supérieur à un terme inférieur. Le terme supérieur reçoit le nom de régissant. Le term inférieur reçoit le nom de subordonné. Ansi dans la phrase "Alfred parle" parle [...] est le régissant et Alfred le subordonné.»

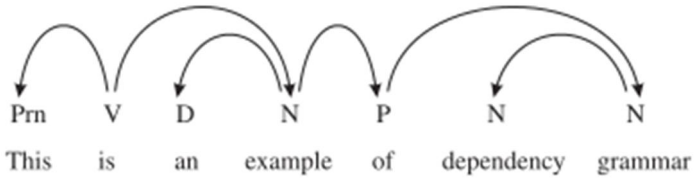

Abb. 3: Ein Dependenz-Baum.

Kurz zusammengefasst hat die Dependenzsyntax folgende distinktive Eigenschaften: a. Es gibt die direkte Kodierung der Struktur des Prädikats; b. Die Dependenzstruktur hängt nicht von der Wortstellung ab; c. Sie passt zu Sprachen mit freier Wortstellung (Latein, Walpiri, etc.); d. Die Expressivität ist eingeschränkt: Jede Dependenzgrammatik hat eine entsprechende kontextfreie Grammatik[15], aber nicht umgekehrt; in einer ungekennzeichneten Dependenzstruktur wird zwischen der Modifizierung der Phrase und des Kopfes nicht unterschieden.

Anhand der zwei oben abgebildeten Theorien erhält man die folgenden Ergebnisse, wenn die Informationen verglichen werden:

1. Die Phrasenstrukturen stellen eindeutig die Phrasen (Konstituenten, oder nicht-terminale Knoten), die strukturellen Kategorien (nicht-terminale Etikette) und einige funktionale Kategorien (grammatikalische Funktionen) dar;

2. Die Dependenzstrukturen bezeichnen die Kopf-abhängigen Verbindungen (direkte Bögen), die funktionalen Kategorien (Bogenetiketten) und einige

[15] Eine kontextfreie Grammatik ist eine Phrasenstrukturgrammatik, die nur kontextfreie Regeln benutzt, d.h. Regeln, für deren Anwendung keine Kontextbedingungen benötigt werden. Die Behandlung dieser Grammatik ist einfach und wird deswegen sowohl für die formale Sprache als auch für die natürliche Sprache stark benutzt.

strukturelle Kategorien (POS).

Laut Delmonte (2014) ist die Dependenzstruktur der Phrasenstruktur vorzuziehen, da sie die zuträglichste der beiden ist: Der Dependenz-Parser zeigt sich starke Flexibilität, Robustheit, Anpassungsfähigkeit, vor allem für die semantische Bereicherung und Verarbeitung.
Es gibt zwei verschiedene Typologien von Dependenz-Parsern: erstens, *rule-based* symbolische Parser, denen die Statistik[16] helfen kann, bzw. reine statistische Parser, die sich eines Models und Klassifizierers bedienen, um zu entscheiden, wie ein Input verarbeitet werden soll (Delmonte 1999, 2000, 2002, 2005);
die zweite Art ist ein auf Phrasenstruktur beruhender probabilistischer Parser mit einer Konversion der syntaktischen Konstituenten in Dependenzen (z.B. *Stanford Parser*, De Marneff et al. 2011).
Im nächsten Absatz wird ParZu als Beispiel der ersten Typologie geschildert.

3.2. ParZu

ParZu[17] ist ein hybrider Dependenzparser für das Deutsche, der externe Hilfsmittel für das Part-Of-Speech Tagging und die morphologische Analyse verwendet. Er ist ein fester und robuster Parser, da er eine von Hand notierte Grammatik (Foth 2005) mit einem Statistikmodul zur Disambiguierung verbindet. Das Statistikmodul wird mit einer Baumbank trainiert.

Er benutzt die gleiche Architektur wie der Pro3Gres[18]-Parser für das Englische (Schneider 2008).

[16]Rein statistische Parser sind als „ill-founded" zu betrachten (Delmonte 2008:129).

[17]Institut für Computerlinguistik, Universität Zürich, 2013. Versionen zu herunterladen: https://github.com/rsennrich/parzu
Online Interface: http://kitt.ifi.uzh.ch/kitt/parzu/

[18]Ein fester und robuster Dependenz-Parser. (vgl. Schneider 2008)

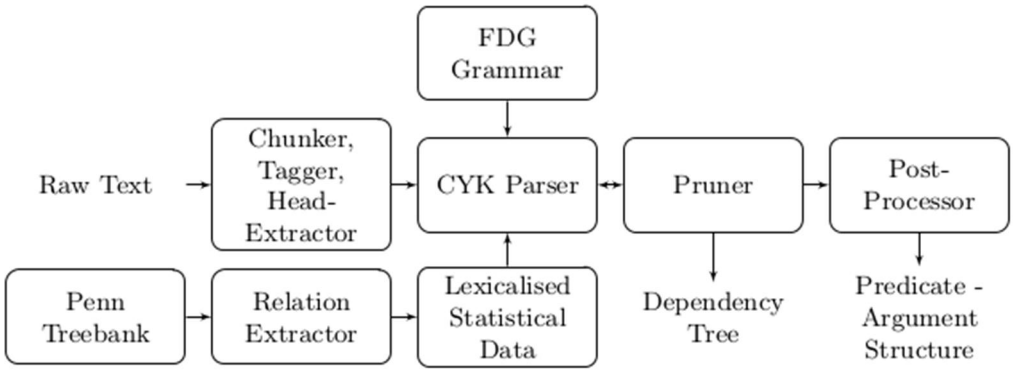

Abb. 4: Die Architektur von Pro3Gres (Sennrich 2009).

Die Anpassung eines vorhandenen Parsers an eine neue Sprache ist ein wichtiger Bereich der Forschung (Nivre et al. 2007). In Hinblick auf die relativ freie Wortstellung des Deutschen müssen die morphologischen und topologischen Regeln in der Grammatik enthalten sein, um die Grenzen der Nominalphrasen sicher zu bestimmen. Aus diesem Grund werden verfügbare linguistische Ressourcen benutzt und mit dem Parser verknüpft.

Zusammenfassend benötigt ParZu drei Komponenten, die nachfolgend kurz erklärt werden:

1. Ein POS-Tagger – TreeTagger;
2. Ein morphologisches Hilfsmittel – GERTWOL;
3. Eine annotierte Baumbank – TüBa-D/Z.

Die in Prolog geschriebene Grammatik ist hauptsächlich nicht lexikalisiert und arbeitet auf der Ebene des POS-Taggings. Um einer Konstituente beispielsweise die Bezeichnung „Subjekt“ zuzuweisen, schränkt die Grammatik die möglichen Labels des Kopfs (des finiten Verbs) und der Dependenten (typischerweise ein Nomen oder Pronomen, aber auch gewisse andere Kategorien wie Zahlen) ein. Der Dependent muss den Kasus Nominativ haben und entweder in Person und Numerus mit dem Verb übereinstimmen oder eine koordinierte Struktur sein. Da jede Wortform mehrfache mögliche morphologische Analysen haben kann, basieren die morphologischen Bedingungen auf der Unifizierung und lassen unterspezifizierte Darstellungen zu.

Bestenfalls wird ein Subjekt für das finite Verb gestattet und manche topologischen Einschränkungen müssen hingenommen werden, wie zum Beispiel eine einzige Konstituente im Vorfeld. Die Regeln stützen sich auf die POS-Informationen – *TreeTagger for POS-Tagging* (Sennrich et al. 2009) - und die morphologischen Kenntnisse – GERTWOL (Haapalainen, Majorin 1995)[19].

Die morphologische Analyse kann die Mehrdeutigkeit verkleinern, wodurch sowohl die Geschwindigkeit des Parsers als auch seine Ergebnisse verbessert werden. Das GERTWOL System lemmatisiert alle Tokens und erzeugt eine mögliche morphologische Analyse dieser Tokens. Diese Lemmatisierung hilft bei dem Problem der zerstreuten Daten im probabilistischen System und die Informationen über den Kasus markieren die grammatikalische Funktion einer Nominalphrase.

Das Statistikmodul zur Disambiguierung modelliert Wort- und Stellungspräferenz und beruht auf der TüBa-D/Z[20], einer handannotierten Baumbank mit ungefähr 65000 Sätzen aus deutschen Zeitungen. Versley (2005) erstellt eine Konversion der Baumbank in ein Dependenz-Format, das der Parser effektiv anwendet. Des Weiteren führt das Statistikmodul eine funktionelle Disambiguierung der deutschen Nominalphrasen basierend auf dem *subcategorization frame* des Verbes durch, kennzeichnet eindeutig die Verbindung der Präpositionalphrasen und Adverbien, und verwendet regelmäßig Wahrscheinlichkeiten, um gewisse Labels zu präferieren, wenn die Grammatik mehrere erlaubt.

```
1     Jetzt  jetzt      ADV   ADV    _                  2     adv
_     _
2     frage  fragen     V     VVFIN  1|Sg|Pres|_        0
      root   _          _
3     ich    ich        PRO   PPER   1|Sg|_|Nom 2       subj  _
_
4     mich   ich        PRO   PRF    1|Sg|Acc           2
      obja   _          _
5     ,      ,          $,    $,     _                  0
      root   _          _
6     wofür  wofür      PWAV  PWAV                _     10    pp
_     _
7     das    die        ART   ART    Def|Neut|Nom|Sg    8     det
_     _
8     Ganze  Ganze      N     NN     Neut|Nom|Sg 10     subj  _
      _
```

[19]Wie in Sennrich (2013) beschrieben wird, kann GERTWOL durch SMOR oder Morphisto ersetzt werden.

[20]TüBa-D/Z ist eine laufende Arbeit einer von Hand notierten Baumbank der Universität Tübingen. Derzeit besteht sie aus 85.358 Sätzen (1.569.916 tokens; 3.444 Artikeln), die aus der Zeitung „die tageszeitung“ herausgenommen wurden.

```
9     gut     gut       ADV    ADJD  Pos|              10
      pred    _     _
10    war     sein       V     VAFIN 3|Sg|Past|Ind     0
      root    _     _
11    .       .         $.     $.    _                 0
      root    _     _
```

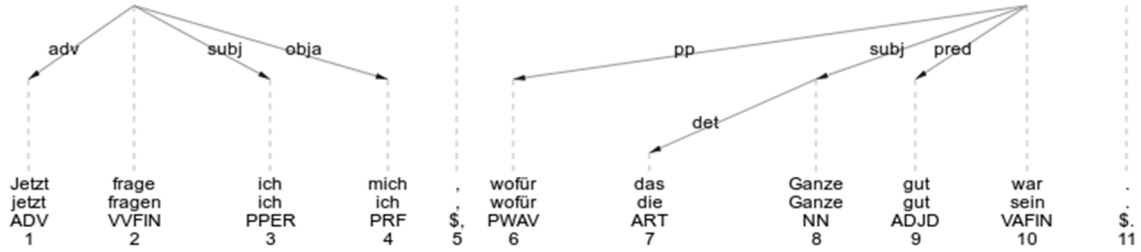

Abb. 5: TüBa-D/Z Baum im CoNLL- und im Dependenz-Format.

Des Weiteren ist noch wichtig, dass ParZu nicht als Akzeptor[21] entwickelt wurde, der möglichst gut zwischen grammatischen und ungrammatischen Sätzen unterscheiden kann, sondern dass er auf Robustheit und gute Qualität für große Textmengen optimiert wurde. Er liefert deshalb auch (Teil-)Analysen für ungrammatische Sätze. Für Sprachlernenden müsste man die Grammatik vermutlich an verschiedenen Stellen anpassen, damit sie ungrammatische Sätze erkennt. Eine Einstellung, die hilfreich wäre, ist `relax_agreement()` in der Datei https://github.com/rsennrich/ParZu/blob/master/core/ParZu_parameters.pl auf `(no)` zu setzen, so dass die morpho-syntaktische Kongruenz strikter geprüft wird. Die Grammatik überprüft die Morphologie aber nur da, wo es für die Parsingqualität der Texten wichtig ist - gewisse Unterscheidungen, z.B. zwischen starker, schwacher und gemischter Flektion bei Adjektiven, werden nicht gemacht.

3.2.1. Constraint-Dependenz-Grammatik

Die von Foth (2005) beschriebene umfassende Constraint-Dependenz-Grammatik des

[21] In der Informatik bedeutet der Begriff Akzeptor „endlicher Automat“ (vgl. Paragraph 3.3.1.), der keine Ausgabe erzeugt. Er prüft die syntaktische Korrektheit eines Inputs (Strings) und kann diese Eingabekette nur akzeptieren oder nicht. (Gasper et al. 1992)

Deutschen soll beliebige Texte analysieren, wobei sie eine geschriebene Eingabe erwartet (Foth 2005:6):

> „1. Einige Regeln verwenden die An- und Abwesenheit von Satzzeichen, um bestimmte Konstruktionen zu erlauben. Ohne Satzzeichen sind diese Regeln wirkungslos oder gar hinderlich.
>
> 2. Eindeutige sprachliche Fehler werden insofern robust verarbeitet, dass es ungeachtet der Eingabe immer mindestens eine mögliche Struktur geben sollte. Jedoch ist dies nicht immer diejenige, die der Sprechintention entsprechen würde. Beispielweise wird ein Kongruenzfehler gewöhnlich nicht zu einer veränderten syntaktischen Struktur führen, ein kategorieändernder Schreibfehler aber sehr wohl („Die Firma will mehr Rechner *verkaufe*"). Daher wird Spontansprache gewöhnlich nicht optimal modelliert."

Als erstes Element dieser Grammatik ist die Syntaxebene zu betrachten. Die syntaktische Satzstruktur wird auf der Ebene `SYN` modelliert und demnach wird die entsprechende Darstellung der Struktur als ein zusammenhängender Baum aufgebaut. Die Satzzeichen `: ; - ()` werden als Unterteilung einer Äußerung oder als Abgrenzung zwischen zwei kompletten Sätzen benutzt. Aus diesem Grund erlauben die Constraints getrennte Bäume eines einzigen Hauptsatzes, der durch solche Zeichen geteilt wird.

Alle Dependenzkanten der Syntaxebene werden durch eine eindeutige Etikette markiert. Ein Unterscheidung findet zwischen Komplementen und Modifikatoren statt: Die erste Kategorie bezieht sich auf alle Arten von Wörtern, die mit ihrem Regenten erscheinen - beispielsweise Subjekte, Objekte, Artikel; die zweite Kategorie stellt die anderen Beziehungen, die niemals obligatorisch sind, dar.

Nachfolgend werden die Label der Komplemente kurz erklärt; alle andere Etiketten befinden sich in Foth (2005): `AUX` bezeichnet die Verbgruppen, wie z.B. Hilfsverben oder Vollverben; die Kante `AVZ` stellt ein abgetrenntes Verb-Präfix zusammen mit seinem finiten Verb dar; `CJ` markiert das Komplement einer koordinierenden (*und*, *oder*) oder vergleichenden Konjunktion (*wie*, *als)*; `DET` bezeichnet den Determinierer eines Nomens, es erfasst bestimmte oder unbestimmte Artikel; `OBJA` ist das Akkusativobjekt, welches das zweite Komplement eines transitives Verbs darstellt; `OBJA2` wird nur bei Verben, die zwei Akkusative erfordern, benutzt (wie z.B. *lehren*, *kosten*, *nennen*); `OBJC` bezeichnet den Objektsatz, d.h. ein finites Verb in

Nebensatzstellung als Komplement eines anderen Verbs (von *dass, ob* oder Fragewörtern eingeleitet); `OBJD` stellt das Dativobjekt - allein oder in Verbindung mit einem Akkusativobjekt - dar; `OBJG` ist das Genitivobjekt; `OBJI` markiert den Objektinfinitiv, ein Infinitiv als Komplement eines anderen Verbs; `OBJP` bezeichnet ein Präpositionalobjekt oder eine Präpositionalphrase, die das Komplement eines Verbs ist; `PN` markiert das Komplement einer Präposition, z.B. Nomen oder Pronomen; `PRED` steht für prädikative Ergänzungen, üblicherweise am Verb *sein* oder auch *werden, wirken, scheinen* - es könnte auch ein prädikatives Adjektiv oder eine Nominalphrase sein; `SUBJ` ist das Subjekt eines finiten Verbs, das immer eine Art von NP ist. Das Wurzelwort eines kompletten Satzes wird vom Label `S` bezeichnet.

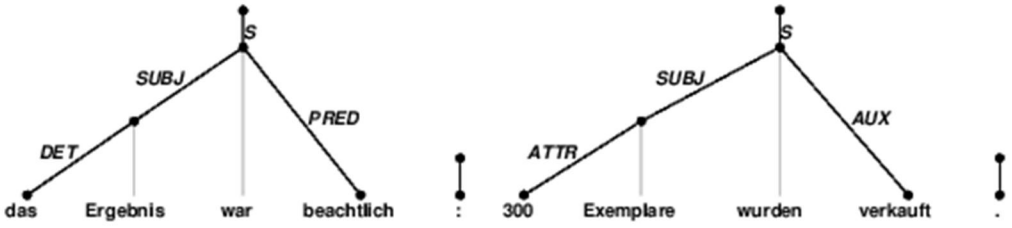

Abb. 6: Die Syntaxebene eines Satzes (Foth 2005).

Das zweite wichtige Element ist die Referenzebene: Auf dieser Ebene werden die Referenzbeziehungen zwischen Pronomen und ihren Antezedenten dargestellt. Nur die Relativpronomen werden beachtet, weshalb die einzige mögliche Kante die vom Relativpronomen zu seinem Antezedens ist.

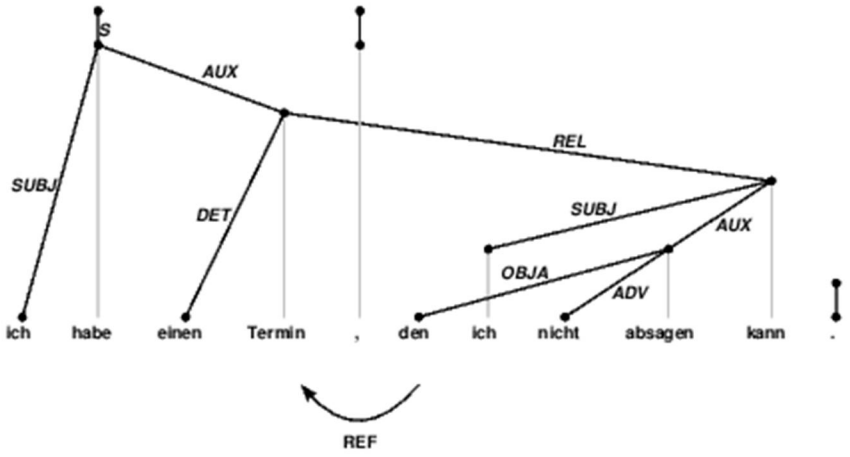

Abb. 7: Die Referenzebene eines Satzes (Foth 2005).

In dieser Grammatik haben die Constraints eine bedeutende Rolle und werden durch ein multiplikatives Bewertungsmaß geschätzt. Wie Foth (2005:33) erklärt, „wählt man *scores* in verschiedenen Größenordnungen, dann sind auch mehrere schwache Fehler immer noch besser als ein viel stärkerer“. Diese Domänen von *scores* treten in Erscheinung:

1.0	nur debug-Constraints
0.9999 bis 0.99	Präferenz-Constraints
0.99 bis 0.5	heuristische Regeln, die öfter verletzt werden
0.5 bis 0.1	minder schwere Fehler
0.1 bis 0.001	schwere Fehler
0.0	Wohlgeformtheitsregeln der Grammatik selbst

Die affinen Constraints werden in Gruppen unterteilt, und auch wenn viele Constraints mehreren Klassen angehören können, werden sie der am besten entsprechenden Kategorie zugewiesen[22]: `POS` bezieht sich auf die Informationen des Part-Of-Speech Taggers, in der Grammatik steht *`pts()`*; `agree` kontrolliert die Kongruenzen oder die Rektionen der Eigenschaften in einer Konstituente, da Verletzungen der Übereinstimmung schwere Fehler verursachen; `category` verlangt die Wortkategorien; `chunker` setzt die Informationen des *chunk parsers* ein; `debug` hat immer Wert 1.0, sodass es nie die Analyse beeinflusst, da dieses Constraint dem Autor zu Diagnosezwecken dient; `dist` (*distance*) zieht die kürzeren Verbindungen vor und ahndet die längeren; `exist` verlangt die Existenz der Komplemente; `init` hat immer den Wert 0.0 und legt Strukturbedingungen fest, die nicht verletzt werden können; `lastresort` wird zur Disambiguierung angewendet; `lexical` kennzeichnet das Verhalten einzelner Worte; `order` begrenzt das Ordnen von verwandten Knoten; `proj` verlangt die Projektivität von Syntaxstrukturen; `root` bestimmt die Wurzelanbindung; `sort` ordnet die verschiedenartigen Restriktionen an; `vorfeld` regelt die Besetzung des Vorfelds.

Die Informationen über die notwendigen und möglichen Bindungen werden größtenteils in jeden Lexikoneintrag eingefügt, d.h. lexikalisiert. Die Werte der lexikalischen Eigenschaften können ein String bzw. eine Reihe, eine Zahl oder Listen sein. In dieser Arbeit werden die Eigenschaften und ihre Bedeutungen nicht behandelt. Letztlich wird die Satzstruktur also durch die Constraints der Grammatik bestimmt;

[22] Andere Constraints und die Makros für Constraintformeln werden in Foth (2005) ausführlich beschrieben.

jedoch können die Beschränkungen Fehler enthalten und unklare Zustände auftreten. Deswegen hat der Autor der Grammatik wichtige Annotationsrichtlinien beigestellt.
Man muss beachten, dass jedes Satzzeichen als ein Wort gilt, und mehrteilige Namen als mehrere Wörter gelten. Apostrophe gehören zum jeweiligen Wort, während Anführungszeichen zu beiden Wörtern gezählt werden. Des Weiteren gelten Komposita mit Bindestrichen als ein Wort. Für den Fall einer fehlerhaften Eingabe, muss die möglichst regelgetreue Analyse ermittelt werden, und sollte ein Wort fehlen, sind die nicht einzuordnenden Bestandteile als Fragment anzusehen. Wenn ein Wort wiederholt auftritt, wird es in den Satz eingebaut, das seinem Regenten näher steht, während das andere als Fragment stehen bleibt. Schreibfehler können darüber hinaus auch die Bedingungen verletzen. In diesem Fall müssen die Bestandteile ebenfalls als Fragment betrachtet werden.
Sowohl die Mehrdeutigkeiten zwischen den syntaktischen Kategorien oder Etiketten als auch die Mehrdeutigkeiten der Struktur und der Morphologie werden vom Autor detailliert berücksichtigt und ihre Verarbeitung erklärt.

In diesem CDG-Formalismus bleiben jedoch einige ungelöste Probleme, da bestimmte Phänomene des Deutschen grundsätzlich im Dependenzformat nicht modellierbar und lösbar sind. Nachfolgend werden die Hauptschwierigkeiten aufgelistet: die ungenügende Behandlung von Ellipsen; das durch `KOKOM` unerkannte Pseudopassiv; die willkürlichen Probleme bei der Tokenisierung.
In Foth (2005) werden alle Problematiken beschrieben und mögliche Lösungen vorgeschlagen.

3.2.2. Bewertung des Parsers

Wie schon erwähnt, wurde ParZu anfangs Pro3GresDE genannt und schon für diese erste Version des Parsers wurde eine Bewertung angefertigt. Die Autoren bestätigten (Sennrich et al. 2009), dass alle gewöhnlichen grammatikalischen Phänomene gut verarbeitet wurden, die singulären – bspw. das Genitivobjekt oder die Nominalphrase im Vokativ – allerdings nicht richtig behandelt wurden. Der Parser wurde zwecks besserer Ergebnisse sowohl mit dem Statistikmodul als auch mit morphologischen Informationen von GERTWOL optimiert. Wenn das Statstikmodul deaktiviert und die

Komplexität der regelbasierten Grammatik erhöht wurde, verringerte sich die Leistung des Parsers. Dies geschieht infolge seltener Dependenzbindungen, wie `PRED` (prädikative Ergänzungen) und `OBJP` (Präpositionalobjekt als Komplement eines Satzes). In beiden Fällen wurden die Etikette der gleichen Labels der Semantikebene unterschieden und die Bindungen fanden nur mit gewissen Verben statt. Wenn eine probabilistische Disambiguierung angewandt wurde, verbesserte sich sowohl der F-score für `PRED` (von 19% auf 67.2%) als auch für `OBJP` (von 20.7% auf 70.2%).
Wenn nur das automatische POS-Tagging benutzt wurde, waren die Ergebnisse deutlich schlechter, weil die Fehler im Tagging-Bereich eine wechselnde Signifikanz hatten. Während der Diskriminierung zwischen Eigennamen und Nomen in dieser Grammatik wenig Bedeutung gegeben wurde, führten die irrtümlich getaggten Verben zur falschen Erwägung aller verbalen Dependenten, die sicher unrichtig verbunden wurden. Der Parser wies auch Schwächen bei der Bildung von Adverbialphrasen auf.

Im Laufe der Zeit wurde der Parser weiter implementiert und eine zweite Bewertung 2013 (Sennrich et al.) vorgenommen. Die Autoren vergleichen zuerst ParZu mit einem statistischen Parser (MaltParser - Nivre 2009), danach mit Pro3gresDE; schließlich schätzen sie den Leistungsverlust, wenn das POS-Tagging geändert wird.
Hier werden die wichtigen Schlussfolgerungen berichtet.
Zusammen mit der Verwendung von MaltOptimizer (Ballestreros, Nivre 2012) wird MaltParser genutzt, ein datengesteuerter Dependenzparser, der verschiedene Algorithmen implementiert. Beide - MaltParser und ParZu - brauchen eine Baumbank - die Version 7 von TüBa-D/Z (mit 1.230.000 Tokens und 65.500 Sätzen).
Die Genauigkeit (*Precision)* und *Recall,* d.h. für wie viele Tokens sowohl der Kopf als auch die Dependenzetikette richtig vorausgesagt werden, werden gemessen, indem die Anzahl korrekter Prognosen entweder mit der Gesamtzahl aller Prognosen oder mit der Anzahl der Bindungen in der Baumbank verglichen wird. Für die Beurteilung werden auch die Tokenisierung und die Teilung der Äußerungen der Baumbank benutzt.
Trotz einer ähnlichen Gesamtleistung haben beide Parser unterschiedliche Stärken und Schwächen. ParZu ist konsistent besser als MaltParser in der funktionalen Disambiguierung der Nominalphrasen, beziehungsweise der Bindungen wie Subjekt, Objekt, Genitivattribut; MaltParser findet indessen mehr Koordinationen, obgleich mit

niedriger Genauigkeit.
Einige f1-Werte (jeweils ParZu und MaltParser): SUBJ 94.5 vs. 90.3; OBJA 87.9 vs. 80.7; OBJD 77.8 vs. 49.6; GMOD 93.8 vs. 88.9.
Wenn das System des POS-Taggings geändert wurde, neigt ParZu zur Ungenauigkeit und er versucht viele partielle Bäume zu generieren.
Unterschiede zwischen dem alten Pro3gresDE und dem neuen ParZu finden sich in den Verbesserungen der Grammatik und des Statistikmoduls zur Disambiguierung.

Eine Abschätzung wurde nur hinsichtlich der Morphologie gemacht. GERTWOL wurde mit zwei Tools - beide basierend auf der SMOR Grammatik (Schmid et al. 2004) - ersetzt. Das erste Tool ist die SMOR Grammatik, die mit dem Lexikon der Universität Stuttgart implementiert wurde. Das zweite, Morphisto (Zielinski, Simon 2009), stellt die SMOR Grammatik mit einem *Open-Source* Lexikon, das von *Creative Commons 3.0 BY-SA Non Commercial license* angeboten wird, dar.
Die SMOR Grammatik erzeugt jedoch keine üblichen Lemmata, sondern abgeleitete Analysen, wie man im Folgenden betrachten kann[23]:

```
> Bewegungen
bewegen<V>ung<SUFF><+NN><Fem><Acc><Pl>
bewegen<V>ung<SUFF><+NN><Fem><Dat><Pl>
bewegen<V>ung<SUFF><+NN><Fem><Gen><Pl>
bewegen<V>ung<SUFF><+NN><Fem><Nom><Pl>
```

Für den Zweck des Parsings lieferten alle drei Hilfsmittel gleichermaßen effiziente Leistungen, wobei Morphisto und SMOR mit +0,1% eine bessere Leistung als GERTWOL erbrachten.
ParZu verlässt sich auf diese externen Analysen, wie die der Morphologie - d. h. um so mehr morphologische Information gefunden wird, desto besser verläuft die Analyse

[23] ParZu benutzt zurzeit das Tool ZMORGE, die Weiterentwicklung von SMOR. (Sennrich et al. 2013b). Hier die Analyse des Wortes „Bewegung“ mit Zmorge:

```
> Bewegungen
Beweg<~>ung<+NN><Fem><Acc><Pl>
Beweg<~>ung<+NN><Fem><Dat><Pl>
Beweg<~>ung<+NN><Fem><Gen><Pl>
Beweg<~>ung<+NN><Fem><Nom><Pl>
Bewegung<+NN><Fem><Acc><Pl>
Bewegung<+NN><Fem><Dat><Pl>
Bewegung<+NN><Fem><Gen><Pl>
Bewegung<+NN><Fem><Nom><Pl>
```

des Satzes.

Es findet sich des Weiteren eine große Leistungsunterscheidung bei den Bindungen von Nominalphrasen, wie zum Beispiel dem Dativobjekt oder Akkusativobjekt.

Außerdem wurde das Tagging geprüft und neue Mittel zur Linderung der Tagging-Einflüsse auf die Leistung des Parsers gesucht. Die eher komplexe Morphologie des Deutschen kann ein Problem für das statistische POS-Tagging darstellen. Die Autoren versuchen, bessere Ergebnisse zu erzielen, und benutzen daher ein *Conditional-Random-Field* Tagger (Lafferty et al. 2001), der morphologische Eigenschaften gebraucht: Seine eigenen morphologischen Eigenschaften wurden durch eine Analyse aus SMOR herausgezogen. Ein Großteil der Probleme resultiert aus der Behandlung unerkannter Wörter.

Während die morphologische Analyse bei dem Tagging unbekannter Wörter hilft, kann der Tag einiger Wortformen durch die lokalen Eigenschaften nicht vorhergesagt werden - beispielsweise der Unterschied zwischen finiten und infiniten Verben (z.B. *erhalten*), und zwischen relativen Pronomen oder Artikeln (z.B. *der*), als auch zwischen Präpositionen und trennbaren Partikeln (z.B. *auf*).

Die Appositionen und Koordinationen bekommen häufig unkorrekte Labels in fehlerhaft getaggten Sätzen. Wenn außerdem ein Verb fälschlicherweise als Nomen getaggt wird, kann die eigentliche Nominalphrase nicht als Subjekt oder Objekt analysiert werden, und sie bekommt stattdessen eine Etikette als Apposition.

Zur Verbesserung der Disambiguierung wurde das *n-best-tagging* benutzt, wodurch kleine Fortschritte bei Pronomen und Artikeln verzeichnet werden konnten. Insgesamt beruhen 30-40% der Fehler auf Verwechslungen zwischen NN (Nomen), NE (Eigennamen) und FM (fremden Wörtern).

Für die Wahl der besten Tag-Sequenz können dieses *n-best-tagging* verwendet und syntaktische Eigenschaften aus ParZu entnommen werden.

Dies erlaubt die Disambiguierung der Tagging-Mehrdeutigkeiten und eine Verbesserung der Genauigkeit des Taggers und der Leistung des Parsers.

Zusammenfassend wurden die Kenntnisse der Morphologie in POS-Tagger durch *Conditional Random Field*-Modelle integriert. Während die Qualität des POS-Taggers für das Parsing wichtig ist, kann andererseits auch der POS-Tagger durch den Parser verbessert werden.

3.3. GI-Tutor

Als Hauptbestandteil dieser Arbeit wird GI-Tutor (GermanItalian Tutor) präsentiert: ein Grammarchecker für das Deutsch, der auf einem *shallow* Parseransatz beruht. Hierbei handelt es sich um einen RTN-Parser (*Recursive Transition Networks*), der zur Verarbeitung des Deutschen in einen ATN-Parser (*Augmented Transition Networks*) umgeformt worden ist.

Dieser Parser wurde schon in der Arbeit von Marino (2000/2001) präsentiert und für die vorliegende Recherche gemäß dem Stand der Technik implementiert und umgewandelt.

Wie schon erwähnt, führt ein Parser eine syntaktische Analyse einer Eingabesequenz durch, um mit Hilfe einer Grammatik und eines Algorithmus die grammatikalische Korrektheit zu überprüfen. Aus einer endlichen Folge von Tokens (Zeichen), die in der Sprachverarbeitung aus Wörtern des Benutzers besteht, setzt sich die Eingabesequenz zusammen. Die hohe Komplexität der natürlichen Sprachen bestimmt die Notwendigkeit starker Einschränkungen bezüglich des Wortschatzes und der grammatikalischen Struktur. Darüber hinaus muss der Zeitfaktor in einem System, das eine starke Interaktion mit dem User fordert, besonders berücksichtigt werden.

Der FSTN-Parser (*Finite State Transition Networks*) stellt eines der ersten erfundenen Parsingverfahren dar, der später in RTN- und ATN-Parser weiter entwickelt wurde. Gazdar et al. (1989:6) definieren RTN-Parser wie folgt:

> „Recursive Transition Networks are one way of specifying grammars. An RTN consists of a collection of networks, each of which is labeled with the name of some syntactic category, or 'part of speech' in traditional parlance. The Networks themselves consist of a collection of states connected by directional arcs labeled with names of syntactic categories. Each network has, in addition, an indication of its initial and final states. The RTN can be interpreted as a collection of maps that permit us to find our way through the grammatical expressions of the language. To determine whether a given string of words is grammatical according to an RTN, it is necessary to find a route through the relevant maps, starting at initial states and ending up at final state. […] An Augmented Transition Network is simply an RTN that has been equipped with a memory and the ability to augment arcs with actions and conditions that make reference to that memory. “

Obwohl FSTN, RTN und ATN auf der gleichen Metapher der Übergangsnetzwerke

basieren, sind FSTN und RTN deklarative Formalismen, während die ATN prozedurale Formalismen (Gazdar et al. 1989) darstellen.

3.3.1. Finite State Automaten

Die endlichen Automaten (*Finite State Automate, FSA*) stellen einen idealisierten, aber einfachen Computer dar; und diese Maschinen sollten zunächst die von Nervenzellen geleistete Verarbeitung emulieren (Kleene 1956). Sie können reguläre Sprachen, die durch reguläre Ausdrücke beschrieben werden können, analysieren (parsen). Daher werden sie derzeit als Muster in der Verarbeitung natürlicher Sprache benutzt.
Die Endlichen Automaten enthalten Zustände – initial und final -, die durch Übergänge miteinander verbunden werden; diese Übergänge sind wiederum mit Symbolen (z.B. Buchstaben oder Wörtern etc.) ausgestattet, die Elemente der zugrunde liegenden Sprache sind. In der Chomsky-Hierarchie[24] entsprechen die Endlichen Automaten den Regulären Ausdrücken.
Sie basieren auf einer *Finite State Grammar* („Grammatik mit endlich vielen Zuständen“, Bußmann 2008:192), die auf der linearen Struktur der Sprache beruht, und die die Bildung einer unendlichen Menge von Sätzen mithilfe einer endlichen Menge von rekursiven Regeln und eines endlichen Wortschatzes ermöglicht.

Ein Übergangsnetzwerk kann die Übergangsfunktion eines FSA genauer veranschaulichen, wobei diese Netzwerke wiederum graphisch dargestellt werden können. *Finite State Transition Networks* sind eine graphische Darstellung der FSA[25] und können wie folgt knapp erläutert werden: Die Eingabekette bewirkt Zustandsübergänge und wird als eine Reihenfolge von Knoten visualisiert, welche ein Label aus der Menge Q oder ein nicht-terminales Symbol der Grammatik bekommen; diese Knoten bezeichnen die verschiedenen Zustände, wobei der Anfangszustand (Si, *initial state*) durch einen kleinen Pfeil indiziert wird und der Endzustand (Sf, *final state*) durch einen Doppelkreis (die Zustände werden durch konzentrische

[24]Die 1956 erstmals von N. Chomsky formulierte Chomsky-Hierarchie bezeichnet eine Einteilung der Sprachtypen: Typ-3 beinhaltet die regulären Sprachen; Typ-2 die kontextfreien Sprachen; Typ-1 die kontextsensitiven; Typ-0 die allgemeinen Regelsprachen. (Carstensen et al. 2010)

[25]Es gibt auch tabellarische Darstellungen.

Doppelkreise geschildert); die Bögen stellen die Übergange dar und werden mit terminalen Symbolen, d.h. lexikalischen Symbolen (Worten), gekennzeichnet, die zeigen, welches Wort sich in der Inputstelle befinden soll, damit sich die Analysefolge als erfolgreich erweist. Die Spitze der Pfeile bestimmen die Richtung der Übergänge.

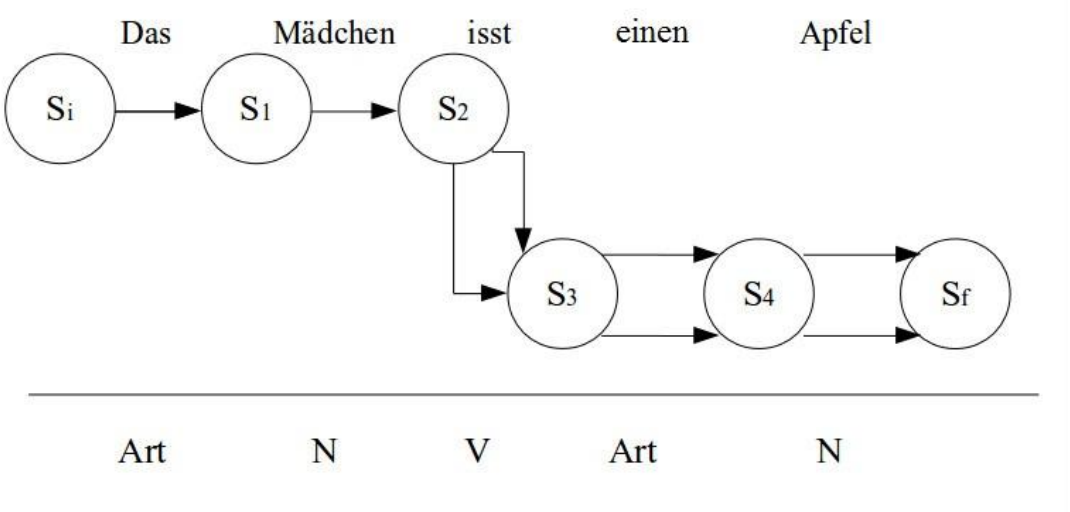

Abb. 8: Darstellung durch FSTN des Satzes „Das Mädchen isst einen Apfel“.

Parallel zum Parsing verläuft die semantische Analyse; eine Ausgabe (vom System FSA) wird jedem Zustandsübergang von FSTN zugeordnet und die semantische Ausgabestruktur wird von den Zustandsübergängen als Ganzes dargestellt.

Wie bereits erwähnt, werden die FSA für die Verarbeitung natürlicher Sprache benutzt, da die Struktur sehr einfach zu implementieren ist. Für das Lösen komplexer Aufgaben sind die Möglichkeiten der FSA jedoch zu begrenzt.

Das Fortschreiten via Bögen, von Knoten zu Knoten, hat zwei wesentliche Aufgaben: erstens, ein Teil der Eingabe kann eingezogen (engl. *consumed*) werden, wenn das Netzwerk zur Erkennung der natürlichsprachigen Texte eingebaut wird; zweitens, Teile des Outputs werden generiert (engl. *produced*), wenn das Netzwerk zur Erzeugung natürlichsprachiger Texte implementiert wird.

Ein Endlicher Automat (nicht deterministisch) wird durch <Q, §, S, F> in einem Alphabet X dargestellt. Hierbei gilt:

a. Q ist eine finite Menge von Zuständen;

b. X ist das Alphabet bestehend aus terminalen Symbolen;

c. § ist die Funktion des definierten Übergangs, *§:QxX → 2^Q* ;

d. S ist der initiale Zustand;

e. F ist die Menge der finiten Zustände.

Ein Zustand stellt immer das Ergebnis aus den Übergangen zwischen den jeweils vorausgehenden Zuständen dar. Jedem Zustand entsprechen Übergänge, denen es durch die Funktion § erlaubt ist, von einem Zustand zum anderen zu „gehen" bzw. diesen zu realisieren.

Diese Gestaltung ist jedoch für die Darstellung und die Generierung von Sätzen unzweckmäßig, da es ihr nicht gelingt, alle Eigenschaften der natürlichen Sprache widerzuspiegeln. Den FSA, bzw. FSTN, mangelt es an Speicherungsmechanismen (*pushdown*), die den Abbruch des Bildungsprozesses eines Satzes erlauben, um mit einem neuen Satz anzufangen; zudem fehlt die Möglichkeit der Speicherung der bereits fertig analysierten Konstruktionen, um diese an den nachfolgenden Teil angleichen zu können. Somit werden Mechanismen von Rekursion direkt in die Übergangsdiagramme eingefügt und ein rekursives Diagramm wird ermöglicht - das heißt ein Diagramm, das sich selbst aufrufen kann.

3.3.2. Recursive Transition Network

Die Unzulänglichkeit des Systems bezüglich bestimmter komplizierter Sprachphänomene, wie z.B. komplexen eingebetteten Sätzen, ist ein Beweis dafür, dass in diesem Bereich verfeinerte Parsing-Strukturen und Vorgehensweisen notwendig sind.

Die *Recursive Transition Networks* (RTN, Rekursives Übergangsnetzwerk) haben, verglichen mit der Struktur von FSTN, noch eine rekursive Funktion, die es Sprachwissenschaftlern erlaubt, rekursive Phänomene natürlicher Sprachen darzustellen. Im Bereich von NLP galten diese rekursiven Übergangsnetzwerke anfangs als ein großer Fortschritt. Darüber hinaus wird der Begriff des indizierten Unternetzwerk (engl. *named subnetwork*) eingeführt: Ein Bogen kann ein Unternetzwerk, in dem die Verarbeitung fortgesetzt werden kann, abrufen und nicht mehr ein bestimmtes Wort oder eine bestimmte Wortfolge, wie zuvor in den FSA. In einem RTN kann ein gesamter Bogenbund, der in den FSTN zur Analyse einer

Äußerung führt, als ein einziges selbstständig indiziertes Netzwerk abgeleitet werden[26]. Wenn nötig, kann dieses selbstständige Netzwerk danach auch in anderen Netzwerken abgefragt werden. Beispielsweise bei der Betrachtung des Satzes „Das Mädchen isst einen Apfel“, erkennt man, dass das Netzwerk S (Satz) die Unternetzwerke NP und VP abrufen muss.
Bezüglich der generativen Fähigkeit entspricht ein rekursives Übergangsnetzwerk einem Automaten mit einem *pushdown* Speicher; es kann als eine Art Verfeinerung dieser *pushdown* Strategie verstanden werden, die jedoch eine bessere Ausdruckseffizienz und flexiblere Struktur hat. Dieser Speicher – *stack* – erlaubt die Funktion der Speicherung. Kurz erläutert: In einem RTN-Parser kann ein Unternetzwerk durchschritten werden, um einen Bogen zu bearbeiten, der mit dem Label des Unternetzwerks gekennzeichnet ist. Das ist jedoch nur möglich, wenn der letzte Knoten, an dem die Verarbeitung momentan unterbrochen ist, abgespeichert bzw. „erinnert wird“, um dann die Verarbeitung im Unternetzwerk fortzusetzen.

In Bezug auf das Beispiel (vgl. Abb. 8) werden die Worte deswegen durch nicht-terminale Symbole, wie z.B. grammatikalische Kategorien, bezeichnet. Eine mögliche einfache Grammatik für den Beispielsatz wäre:

```
S → NP V NP
NP → ART N
```

[26]Gazdar et al. (1989) sagten hierzu: “The idea is that if we have commonly used bunch of arcs, we can express this abstraction by making it into a self-contained, named network. This network can then be referenced by its name in a network that needs it, rather than having to appear expanded out in every place.”

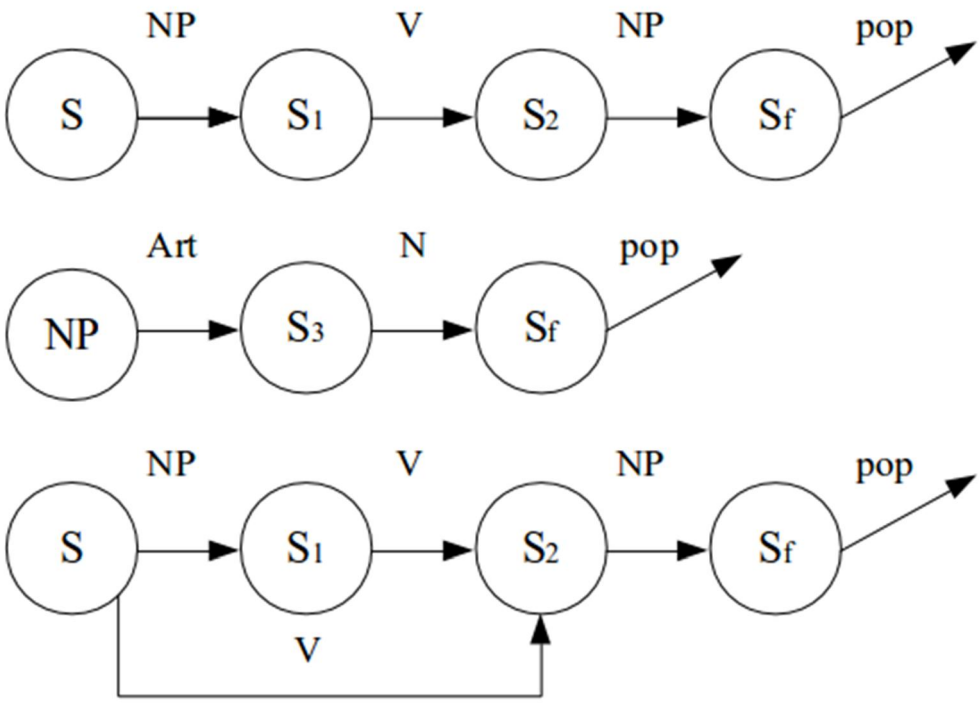

Abb. 9: Zustandsgraph des Beispielsatzes „Das Mädchen isst einen Apfel."

Wenn ein nicht-terminales Symbol getroffen wird, wird die Verarbeitung an dieser Stelle kurzzeitig abgebrochen und das entsprechende Unternetzwerk gesucht, um zunächst dort die Verarbeitung fortzusetzen. Wenn hier der Endzustand S_f erreicht ist, kann dann die Verarbeitung an dem ursprünglichen Knoten fortschreiten. Zwei wichtige Hauptoperationen werden für die Speicherung der Informationen - *pushdown store* - benötigt: Durch die Operation PUSH werden Informationen auf dem Stapel (*stack*) ergänzt und ein Unternetzwerk wird zwischenzeitlich aufgerufen; durch die Operation POP werden die zuletzt eingefügten Informationen aus dem Stapel weggenommen, wenn der Endzustand in dem Unternetzwerk erzielt worden ist. So kann die Verarbeitung in dem vorhergehenden Netzwerk fortgesetzt werden.

Auf diese Weise werden die Namen der Knoten und der Netzwerke gespeichert. Die Analyse einer Eingabe gilt als erfolgreich, wenn der Endzustand S_f in dem *top-level* Netzwerk erreicht werden konnte und somit die ganze Eingabesequenz vollständig analysiert wurde.

Die Eigenschaft der Rekursivität bedeutet, dass das gleiche Netzwerk auf jedem Bogen wieder erneut aufgerufen werden kann. Diese Methode erlaubt, komplizierte

Sprachphänomene – bzw. die Einbettung (S NP V S) – automatisch zu bearbeiten und zu analysieren. Dieses Muster von RTN zeigt gleichwohl einige Einschränkungen, wenn kontextabhängige Sätze (passive Äußerungen, Fragesätze, bejahende Sätze, etc.) ausgedrückt und analysiert werden sollen. Die RTN sind deshalb ungeeignet, wenn die typischen Feinheiten der natürlichen Sprache abgebildet werden sollen.

3.3.3. Augmented Transition Network

Wie soeben dargelegt, sind RTN deutlich leistungsfähiger als FSA – auch durch ihre Fähigkeit Rekursivität darstellen zu können. Dennoch liegen noch zwei bedeutsame Schwierigkeiten vor: RTN analysieren fehlerhafte Sätze, ohne deren Ungrammatikalität zu erkennen, und sie erzeugen keine Struktur des Inputs am Ende der Analyse. Ein ATN-Parser (Woods 1983) weist diesbezüglich gewisse technische Eigenschaften und Umwandlungen auf, um die obengenannten Mängel zu beheben.

Nach Woods Studie (1970) wurden auch die ATN (*Augmented Transition Network* – Erweiterte Übergangsnetzwerk) am Anfang als Tool für die Analyse der natürlichen Sprache, und danach für die Generierung der Sprache, angewandt. Ursprünglich entwickelt wurden sie als implementierbares Gegenmodell zur generativen Transformationsgrammatik[27].

Statt Phrasenstrukturregeln benutzen diese Parser eine äquivalente Menge von Finiten Automaten (vgl. Paragraph 3.3.1.), die ebenfalls die Eigenschaft der Rekursion aufweisen (vgl. Paragraph 3.3.2.). Sie können somit ebenfalls untergeordnete Netzwerke aufrufen. Die sogenannten Augmentationen (d.h. Zusätze) enthalten komplexere Speicheroperationen auf den Bögen, die erlauben, die Robustheit zu prüfen.

Wie Winograd (1983) bestätigt,

> „The augmentation centers around the addition of conditions and actions associated with the arcs of a network. Conditions restrict the circumstances under which an arc can be taken, while actions perform feature-marking and structure-building operations. Conditions and actions make use of registers for roles and features, associated with the nodes of the parse tree being constructed."

[27] Jede generative Grammatik, die Transformationsregeln benutzt, ist eine Transformationsgrammatik (Bußmann 2008).

Wenn ein RTN-Parser genommen und weitere Verbesserungen implementiert werden, ergibt sich daraus ein ATN-Parser. Diese Verbesserungen betreffen erstens das Einführen von Listen (*registers*), die eine provisorische Speicherung der Informationen, z.B. der erkannten Strukturen oder grammatikalischen Kategorien, gestatten. Auf diese Weise werden Informationen bewahrt, die durch den Zustand des Netzwerks implizit gespeichert werden sollen.

Erneut soll das Beispiel (vgl. Abb. 8) „Das Mädchen isst einen Apfel“ betrachtet werden. Nach allen drei Übergängen werden drei Listen gespeichert: die erste mit dem Output NP als Subjekt; die zweite mit dem Output V als Prädikat und die dritte mit NP als Objekt. Diese drei Kennzeichnungen können mit zwei verschiedenen Strukturen geschildert werden, die entweder eine einfache indizierte Klammerung (Abb. 10) oder eine Baumdarstellung (Abb. 11) sein können.

Parse:

```
sogg-[sn-[art-[das],n-[maedchen]]]
ibar-[
     vt-[isst]
svt-[
     ogg-[sn-[art-[einen],n-[apfel]]]
```

Tags:

```
[das-[art,deit,rel],maedchen-[n],isst-[vt],einen-[art],apfel-
[n],. - [punto]]
```

Abb. 10: Indizierte Klammerung.

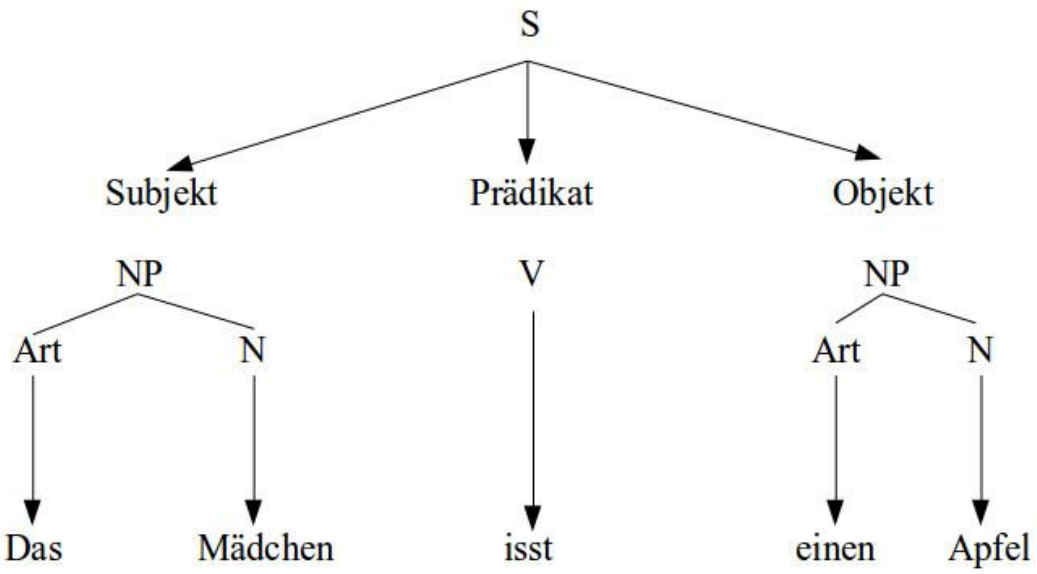

Abb. 11: Mögliche Baumdarstellung des Satzes „Das Mädchen isst einen Apfel."

Die Bögen werden nicht nur mit grammatikalischen Kategorien und syntaktischen Strukturen, sondern auch mit bestimmten Kontrollen (*tests*) in Zusammenhang gebracht. Wenn sich der Bogen V beispielsweise mit dem Test der Übereinstimmung NP-V verbindet, werden die für die Übereinstimmung NP-V fehlerhaften Sätze sofort automatisch beseitigt.

Darüber hinaus können beliebige Aktionen mit den Übergangsbedingungen zwischen den Zuständen durchgeführt werden, um beispielsweise Strukturbäume oder semantische Darstellungen aufbauen zu können.

Verschiedene Arten von Bögen, die unterschiedliche Funktionen haben, werden folgend kurz beschrieben:

a. Der Bogen „CAT" wird verarbeitet, wenn er die gleiche Kategorie des zu analysierenden Wortes trägt;

b. Der Bogen „PUSH" bricht die aktuelle Verarbeitung ab und ruft ein Unternetzwerk auf (vgl. Paragraph 3.3.2.), in dem die Verarbeitung bis zur Erreichung des Endzustandes weitergeführt wird;

c. Umgekehrt wird mit dem Bogen „POP" die Verarbeitung des „PUSH"-Bogens fortgesetzt, nachdem er das Ende des Unternetzwerks erreicht hat, da er normalerweise das Ende eines Bogens darstellt;

d. Mit der Funktion „JUMP" findet ein Übergang statt, ohne das eingegebene

Symbol der Analyse zu unterziehen.

3.3.3.1. Der Bau von GI-Tutor

In den folgenden Absätzen werden die grundlegende Struktur von GI-Tutor und der Aufbau dieses RTN/ATN-Parsers vorgestellt.
Ferner werden zusätzliche Strukturen vorgestellt, die diesem Parser zugrunde liegen, um die Aufgabe der Fehlererkennung und -analyse vollständig erfüllen zu können. Drei Einheiten des Programms haben eine entscheidende Rolle: erstens, die Regeln des Parsers; zweitens, der Algorithmus zur Fehleranalyse; drittens, der Algorithmus zur Fehlermeldung. Nachfolgend werden sie auch mittels Beispielen genauer erläutert. Der komplette Output des Parsers besteht aus der in Klammern gesetzten annotierten Eingabekette, die zu einer hohen grammatikalischen funktionsfähigen Prozedur weitergeht. Dies ist die Überführung einer LFG[28]-basierte C-Struktur in eine F-Struktur (Bresnan 1982), die drei Zwecken dient: die Eigenschaften des Kopfes zu berechnen; die Kongruenz zu berechnen; die LFG-grammatikalischen Prinzipien der Kohärenz und Eindeutigkeit (*consistency*) einzuführen, damit die Numeri und die Typen der Argumenten von der lexikalischen Form des Prädikats beschränkt werden.

Unter dem Begriff „Regeln des Parsers“ sind alle Regeln, denen sich der Algorithmus für die Satzanalyse bedient, zu verstehen. Sie werden benötigt, um die syntaktischen Kategorien der einzelnen Konstituenten zu erkennen und die Wohlgeformtheitsbedingungen der gegebenen Aussagen zu überprüfen. Der Parser wurde so aufgebaut, dass jeder Satz – richtig oder fehlerhaft ausgedrückt – einen Output bekommt. Obwohl der Parser *shallow* ist, bekommen die Lernenden ebenso detaillierte Informationen über ihre Fehler.
In einem in Prolog geschriebenen Programm ist die Zuordnung der Regeln sehr grundlegend und deren Reihenfolge bzw. Aufbau folgt nur teilweise den Regeln der deutschen Grammatik, da so die Robustheit des Parsers nicht unterbewertet wird. Als Erklärung dieser Unvollständigkeit muss die Fehleranalyse durch den Parser

[28]Lexikalisch-funktionale Grammatik (Lexical Functional Grammar): Für die Erklärung dieser Grammatik siehe: Bresnan und Kaplan 1982; Johnson 1988.

verstanden werden: Diese wäre ohne die Integration „falscher“ Grammatikregeln nicht möglich, da sonst der Parser bei jeder Konfrontation mit ungrammatikalischen Sätzen die Analyse abbrechen würde, anstatt sie zu beenden und eine Fehlerrückmeldung zu geben.
Diese „falschen“ Regeln sind basierend auf den deutschen Äußerungen italienischer Muttersprachler formuliert, und erlauben letztlich eine vollständige Analyse fehlerhafter Sätze. Diese Sätze sind in einem Korpus (vgl. Kapitel 4) gesammelt. Dadurch ist also die Fehlererkennung daraufhin die Fehleranalyse selbst ermöglicht.

Die erste Phase der Analyse ist durch den Algorithmus zur Fehleranalyse charakterisiert, der aus einer finiten Menge von Operationen besteht und die oben genannten Regeln für das Analysieren benutzt. Trotz der zahlreichen Lexikoneinträge vollzieht er eine flache (*shallow*[29]) Analyse, das heißt eine oberflächliche Erkennung der Einheiten des Satzes, und zusätzlich eine Nachprüfung der Grammatikalität des Satzes. Die Fehler können in diesem Parsingverfahren entdeckt werden, jedoch zeigt der Parser keine besondere Rückmeldung auf die Inkorrektheiten. Der Output beinhaltet einfach die annotierte Eingabekette, die durch eine einfache indizierte Klammerung dargestellt wird.

```
1. ?- rtn_parse(f,1,[ich, fahre, in,
'Frankreich', .],[],Parse,Tags).

[ich-[pers,pron],fahre-[vin,vinimp],in-[p],Frankreich-[np],. -
[punto]]

Parse = [sogg-[sn-[pers-[ich]]], ibar-[vin-[fahre], svin-[sp-
[... - ...|...]]], punto-['.']],

Tags = [pers-[ich], vin-[fahre], p-[in], np-['Frankreich'],
punto-['.']].


2. ?- rtn_parse(cp,1,[als, die, vorlesungen, zu, ende, war,
wollte, ich, nicht, nach, hause,
zurueckkommen, .],[],Parse,Tags).
```

[29] Parser können entweder *deep* oder *shallow* anlysieren: Ein deep Parser, d.h. „tief“ oder „tiefgründig“, führt eine vollständige Sprachanalyse auf allen wichtigen Wissensebenen durch; ein shallow Parser, d.h. „flach“ oder „oberflächlich“, macht eine verminderte und partielle Analyse (Carstensen 2012). Ebenso können die Parser fehlersensitiv oder fehlerrobust sein (Schwind 1988): Beim fehlersensitiven Parsen kann ein Satz trotz eines Fehlers analysiert werden, jedoch wird dieser Fehler erkannt und erklärt. Dagegen wird ein Fehler beim fehlerrobusten parsen nicht untersucht und dargestellt.

```
[als-[cosu,p],die-[art,rel],vorlesungen-[n],zu-
[in,p,pt2,tren],ende-[n],war-[ause,vc],wollte-[vsupp],ich-
[pers,pron],nicht-[neg],nach-[p,tren],hause-[n],zurueckkommen-
[viin,vin],. - [punto]]

Parse = [fs-[cosu-[als], fyesno-[sogg2-[sn-[...|...]], ogg-
[... - ...], ibar2-[...]]], yesno-[vsupp-[wollte], fyesno-
[sogg2-[... - ...], neg-[...], ... - ...|...]]],

Tags = [cosu-[als], art-[die], n-[vorlesungen], in-[zu], n-
[ende], ause-[war], vsupp-[wollte], pers-[...], ...
- ...|...].
```

3. `?- rtn_parse(cp,1,[wenn, mutti, das, geld, mir, gibt, kann, ich, neue, schuhe, kaufen, .], [], Parse, Tags).`

```
[wenn-[cosu],mutti-[n],das-[art,deit,rel],geld-[n],mir-
[clitdat,pron],gibt-[vt],kann-[vsupp],ich-[pers,pron],neue-
[ag],schuhe-[n],kaufen-[vit,vt],. - [punto]]

Parse = [fs-[cosu-[wenn], fyesno-[sogg2-[sn-[...]], ogg-[...
- ...], ogg-[...], ... - ...]], yesno-[vsupp-[kann], fyesno-
[sogg2-[... - ...], ogg-[...], ... - ...]], punto-['.']],

Tags = [cosu-[wenn], n-[mutti], art-[das], n-[geld], clitdat-
[mir], vt-[gibt], vsupp-[kann], pers-[...], ... - ...|...].
```

4. `?- rtn_parse(cp,1,[das, interrail, war, eine, lehrreiche, erfahrung, .] ,[], Parse, Tags).`

```
[das-[art,deit,rel],interrail-[n],war-[ause,vc],eine-[art],
lehrreiche-[ag], erfahrung-[n],. - [punto]]

Parse = [f-[sogg-[sn-[art-[das], n-[...]]], ibar-[vc-[war],
svc-[... - ...]], punto-['.']]],

Tags = [art-[das], n-[interrail], vc-[war], art-[eine], ag-
[lehrreiche], n-[erfahrung], punto-['.']].
```

Die zweite Phase der Analyse besteht aus dem Algorithmus zur Fehlermeldung: Die Konstituenten werden kontrolliert, überprüft und gegebenenfalls automatisch korrigiert. Die morpho-syntaktischen Eigenschaften werden aus dem Lexikon extrapoliert und die Grammatikalitäts- und Wohlgeformtheitsbedingungen der fehlerhaften Konstituenten werden hiermit geprüft, um die Fehler zu erkennen.
Mit dem Begriff „morpho-syntaktischen Eigenschaften“ sind die folgenden *Features* gemeint: Genus, Numerus, Person und Kasus von Adjektiven, Pronomina und Artikeln, Modus, Tempus, Person, Numerus und Valenz von Verben, Rektion von Präpositionen, usw.

Diese Angaben sind im Algorithmus durch den Eintrag *morfo* dargestellt; *mem_feats* speichert diese Eigenschaften, die durch *check_accordo* passend kontrolliert werden. Die Fehlermeldung – das Feedback - kann durch den Eintrag *messaggio* im Outputfenster erzeugt werden. In den folgenden Beispielen können sowohl die funktionalen Strukturen als auch die Fehlermeldung beobachtet werden.

```
5. ?- parse_funct(cp,[die, auswahl, war, nicht, so, gross, .],
Out).
die auswahl war nicht so gross .

f-[feats- (fs3-[acc,nom]),sogg-[die,auswahl],pred sein, nicht,
so, gross]

Out = [f-[f-[feats- (fs3-[acc, nom]), sogg-[die, auswahl],
pred-sein, nicht, so|...]]].
```

```
6. ?- parse_funct(cp,[viele, firmen, brauchten, ihre,
kenntnisse, .] , Out).
viele firmen brauchten ihre kenntnisse .

Caso del Complemento giusto

f-[feats- (fp3-[acc,nom]),sogg-[viele,firmen],pred-
brauchen,ogg-[ihre,kenntnisse]]

Out = [f-[f-[feats- (fp3-[acc, nom]), sogg-[viele, firmen],
pred-brauchen, ogg-[...|...]]]].
```

```
7. ?- parse_funct(cp,[die, kathedrale, wurde, fast, voellig,
zerstoert, im, jahr, 1945, .], Out).
die kathedrale wurde fast voellig zerstoert im jahr 1945 .

Posizione Verbo Participio Passato Sbagliata
predicato =  zerstoeren
costituente =  svpass
Caso del Complemento Obliquo giusto

f-[feats- (fs3-[acc,nom]),sogg-[die,kathedrale],voice-
pass,pred-zerstoeren,sp-[im,jahr,1945]]

Out = [f-[f-[feats- (fs3-[acc, nom]), sogg-[die, kathedrale],
voice-pass, pred-zerstoeren, ... - ...]]].
```

```
8. ?- parse_funct(cp, [man, lernen, franzoesisch, in, der,
schule, .] ,Out).
man lernen franzoesisch in der schule .

Accordo Soggetto-Predicato sbagliato
lernen = terza persona plurale
soggetto =  terza persona singolare
Caso del Complemento Obliquo giusto
```

```
Caso del Complemento giusto

f-[feats- (mfs3-[nom]),sogg-[man],pred-lernen,ogg-
[franzoesisch],sp-[in,der,schule]]

Out = [f-[f-[feats- (mfs3-[nom]), sogg-[man], pred-lernen,
ogg-[...], ... - ...]]].
```

Zusammenfassend gestatten die drei hier knapp vorgestellten Einheiten, dass die Analyse trotz vorhandener Fehler, bis zum Ende durchgeführt wird. Die Fehlerkorrekturen werden nicht erzeugt, da zu diesem Thema verschiedene pädagogische Ansätze existieren und Studien gezeigt haben, dass Fehlererklärungen effektiver sind, als reine Korrekturen.

3.3.3.2. Die Verbesserungen

Um die Ergebnisse der Satzanalyse zu verbessern, wurde der Parser überarbeitet und implementiert. Zuerst wurde das Trainingskorpus vergrößert und neue Worteinträge wurden in die Lexikondatei eingefügt. Die Wörter, die Rechtschreibfehler beinhalten, wurden anfangs als Ausnahme im Wörterbuch betrachtet. Jedoch haben diese Einträge dem Parser Schwierigkeiten in der Strukturanalyse bereitet, wurden daher letztlich wieder aus dem Lexikon gelöscht.

Darüber hinaus wurde das Lexikon mit einer angepassten Version des JWCDG (*Constraint Dependency Grammar*) Lexikons ausgedehnt. Diese neuen Lexikondateien wurden einer Internetseite der Universität Hamburg[30] entnommen. Diese beinhalten Verben, Adjektive, Nomen und Eigennamen, die nach den Regeln einer Constraint-Dependenz-Grammatik kategorisiert und mit dem Stuttgart Tübingen Tagset (STTS) annotiert wurden. Jedes File enthält 200.000 Einträge circa.

Da GI-Tutor ein Konstituenz-Parser ist und in Prolog funktionieren sollte, wurden diese Lexikondateien sorgfältig verarbeitet. Die Dependenzetiketten wurden durch Konstituenzetiketten des GI-Tutor Tagsets (vgl. Anhang 1) ersetzt. Diese beinhalten 85 Etikette, die wichtige Eigenschaften für den Parser kodieren bzw. verschlüsseln: die Transitivität, die Modalität, die Klasse der Auxiliarverben, semantische Kategorien

[30] https://nats-www.informatik.uni-hamburg.de/CDG/DownloadPage#Download

– wie z.B. Farbe und Menschlichkeit - sowie die wichtigsten Kategorien für Nomen. Die Etikette werden durch eine statistische und syntaktische Prozedur disambiguiert, die für besondere Klassen der Zweideutigkeit eingerichtet wird. In einigen speziellen Fällen werden organisierte Finite State Automaten verwendet. Der Output des *Disambiguators*[31] besteht aus den teilweise disambiguierten Eingabeketten, die vom *shallow* Parser verarbeitet wurden. Das Tagging der Partikeln – bspw. *zu*, *um* - stellte am Anfang technische Probleme dar, weil der Parser oft die falsche Etikette ausgegeben hat.

Das folgende Beispiel stellt das Tagging und die Analyse des Satzes „*Mein Opa bringt ihr sie“ dar:

```
?- parsefunct(o1).
mein opa bringt ihr sie .

##############################################################
                TAGGING DELLA FRASE DA ANALIZZARE
##############################################################

[mein-[poss],opa-[n],bringt-[vt],ihr-
[clitdat,pers,poss,pron],sie-[clitac,pers,pron],. - [punto]]

##############################################################
                DISAMBIGUAZIONE DEI TAGS AMBIGUI
##############################################################

[mein-[poss],opa-[n],bringt-[vt],ihr-
[clitdat,pers,poss,pron],sie-[clitac,pers,pron],. - [punto]]

##############################################################
                STRUTTURA IN COSTITUENTI SINTATTICI:
##############################################################
```

31 Der Begriff kommt aus dem Bereich der Computerwissenschaft. *Disambiguator* ist eine Applikation der Verarbeitung natürlicher Sprachen, die versucht, durch eine linguistische Kontextanalyse die beabsichtigte Bedeutung eines Wortes oder Satzes zu bestimmen. http://www.thefreedictionary.com/disambiguator

```
sogg-[sn-[poss-[mein],n-[opa]]]
ibar-[
     vt-[bringt]
svt-[
     ogg-[sn-[clitdat-[ihr]]]
     ogg1-[snsempl-[pron-[sie]]]
          punto-[.]
```

Abb. 12: Darstellung der Analyse mit GI-Tutor.

In Prolog ist die Reihenfolge der Regeln für die korrekte Arbeitsweise des Parsers enorm wichtig. Aus diesem Grund wurden die Regeln neu angeordnet und es wurden neue Regeln gebildet, um beispielsweise die Relativsätze besser analysieren zu können (vgl. Abschnitt 4.4.).
Eine zusätzlich neue Regel dient, den Kasus und die Stellung der Akkusativ- und Dativergänzungen zu kontrollieren, nicht nur wenn sie als Nominalphrase ausgedrückt sind, sondern auch wenn sie als Pronomina artikuliert werden.

Die syntaktische Struktur stammt von den flachen Strukturen durch zwei einfache Umschreibungsoperationen: die Löschung (*deletion*) und die Umstrukturierung (*restructuring*). Für die Bildung der syntaktischen Konstituenten werden die folgenden Kriterien befolgt:

1. Sowohl die deutsche als auch die italienische syntaktische Struktur werden angenommen;
2. Die Konstituenz soll die Wiederherstellung (*recovery*) der Fehler sowohl in den Konstituenten als auch in der höheren strukturellen Ebene erlauben;
3. Das konjugierte Verb wird auf eine besondere Weise behandelt: Wenn es sich am Ende des Satzes befindet, gehört es zu einer getrennten ibar-Konstituente - bzw. IBAR2 – und es erzeugt die Bildung einer besonderen IP-Konstituente – bzw. FYESNO – für die *aux-to-comp*-Strukturen und die Strukturen, welche die Inversion verlangen; ansonsten wird das Verb nach den italienischen Grammatikregeln behandelt.

Ungefähr 700 fehlerhafte Sätze wurden gesammelt, um ein Trainingskorpus zu bauen, und jeder Satz wurde analysiert: Zuerst wurde der Fehler gekennzeichnet und danach

wurden die Fehler nach den Hauptsprachebenen eingruppiert, um die Klassifizierung zu vereinfachen und diese im Vergleich zu Marinos (2000/2001) intuitiver zu gestalten. Das Korpus und die neue Gliederung werden im Kapitel 4 erklärt.

Die für Italiener typischen Fehler werden mithin beobachtet: Kongruenzfehler in der Nominalphrase; falsche Stellung der Klitika (Pronomina); Kongruenzfehler zwischen Subjekt und Verb; Weglassung der Subjektpronomina; falsche Stellung des konjugierten Verbs in den Hauptsätzen, Nebensätzen, oder beigeordneten Sätzen; und falsche Kasusbildung.

Das vorherige Beispiel (vgl. Abb. 12) stellt diesen Prozess dar:

```
##############################################################
FEEDBACK DEL CORRETTORE GRAMMATICALE
##############################################################
Caso del complemento indiretto giusto
Caso del complemento diretto sbagliato
oggetto diretto = caso accusativo [sie]
caso [acc]
Posizione oggetti invertita
oggetto diretto = ihr
oggetto indiretto = [sie]

f-[feats-     (ms3-[acc,nom]),sogg-[mein,opa],pred-bringen,ogg-
ihr,ogg1-sie]
```

In diesem Fall erzeugt der Parser zwei Fehlermeldungen: Die erste zeigt die falsche Kasusbildung, da das Pronomen „sie“ den Kasus Dativ statt Akkusativ verlangt; die zweite Meldung betrifft die Stellung des Dativklitikons „ihr“, die sich nach dem Akkusativklitikon finden sollte.

Damit die Fehler immer erkannt werden, müssen sämtliche morphologischen und syntaktischen Wortinformationen zur Verfügung stehen. Die Rückmeldungen des Parsers werden auf Italienisch formuliert, da das System für Lernende mit italienischer Muttersprache gedacht ist. Des Weiteren werden neue Feedbacks eingefügt, damit das System dem Lernenden stets eine geeignete Meldung geben kann.

4. Das maschinelle Analyseverfahren

In diesem Kapitel werden folgende grundlegende Begriffe dieser Arbeit behandelt und erklärt: der Fehler, die Fehleranalyse, und die Wichtigkeit des Feedbacks. Danach wird ein kleines Korpus von fehlerhaften Sätzen untersucht und die Fehler werden methodisch klassifiziert. Mithilfe der im Kapitel 3 vorgestellten Parser wurden diese Äußerungen analysiert, und diese Ergebnisse sollen hier ausführlich präsentiert werden.
Zusätzlich werden die drei verschiedenen Typen von Übungen, die in GI-Tutor eingebaut sind, erklärt.

Allgemein betrachtet, sollten Fehler nicht als ein Zeichen von Misserfolg betrachtet werden, da sie Einblick in die Entwicklung und den Status des Lernens – wie auch des Lernenden – geben.

Mit dem Begriff Fehleranalyse - auch Fehlerlinguistik genannt (Bußmann 2008:189) - werden sowohl die methodische Beschreibung der Fehler in der Fremdsprachdidaktik als auch die Einschätzung und Therapie dergleichen in der Fehlerpädagogik bezeichnet. Im Rahmen dieser Arbeit wird nur der erste Aspekt bearbeitet.

Der Begriff Feedback bezeichnet eine besondere Antwort und Reaktion auf einen Fehler, die jedoch keine besondere Beschreibung oder Diagnose bietet (James 1998). Nach vielen SFLLF-Theoretikern und CALL-Experten müsste der Terminus allerdings auf folgende Bedeutung erweitert werden: Das Feedback beinhaltet jegliche Information, die der Lernende nach einer Aufgabe erhält.

4.1. Die Bezeichnung der Fehler

Beim Erlernen einer Sprache (L2, Zielsprache) kann der Lernende sich auf das vorhandene kognitive System der Muttersprache (L1) stützen, wobei dies auch Nachteile mit sich bringen kann.

Fehler geben Einblick in den Prozess der Textproduktion, da sie eine Abweichung von den Regeln der Zielsprache darstellen (Heift, Schulze 2007). Sie spiegeln die linguistischen Normen, welche die Lernenden von ihrer Muttersprache

auf die L2 übertragen, wider.

In der Fremdsprachdidaktik werden zahlreiche Fehler identifiziert, die nach verschiedenen Faktoren und Aspekten kategorisiert werden können. Die wichtigsten können wie folgt eingeteilt werden:

1. nach den Hauptfertigkeiten der Sprachdidaktik, d.h. Sprechen, Hören, Schreiben, Lesen;
2. nach den linguistischen Beschreibungsebenen, also Orthographie und Graphemik, Phonetik und Phonologie, Morphologie, Syntax, Lexikon (Wortschatz), Phraseologie, Stilistik;
3. nach der Fehlerform, wie bspw. Auslassung, Einfügung, Kontamination, Vertauschung, usw.;
4. nach dem Fehlertyp, z.B. Normfehler, systematische Fehler (*error*), oder Flüchtigkeitsfehler (*mistake*), als auch Fehler auf der Kompetenz-Performanz-Ebene;
5. nach der Ursache, wie z.B. Interferenz, *Interlanguage*, Interimssprache, usw.

Der vierte Punkt bezeichnet hierbei eine primäre Unterteilung der Fehler, die die Unterscheidung zwischen *errors* und *mistakes* (Corder 1974) betrifft: *errors* ergeben sich aus dem Mangel an Kompetenz (*competence*) des Lernenden in der Zielsprache; *mistakes* ergeben sich hingegen aus der Performanz und vermehren sich unter ungünstigen Bedingungen wie Stress oder Müdigkeit. Beide sind als Abweichungen von der Norm zu betrachten und werden als solche von dem Parser erkannt. Der größte Unterschied findet sich dann in der Art des Feedbacks[32], das der Lernende bekommt: Der erste Typ von Fehler erfordert eine Hilfe, damit die Kompetenz in dem entsprechenden Bereich verbessert wird; der zweite kann jedoch vom Lernenden selbst korrigiert werden, sodass das Feedback keine detaillierte Erklärung des Fehlers liefern muss.

In Punkt 5 wird das Thema der *Interlanguage* (dt. Interlingua) angeführt, deren Verstehen durch die Fehleranalyse vertieft wird. Die Interimssprache ist eine Varietät der Zielsprache zu einem spezifischen Zeitpunkt (Ellis 1994); dies bedeutet, dass sich

[32]Der Begriff wird im Abschnitt 4.2. erläutert.

die Lernersprache an einem Punkt zwischen der Muttersprache und der Zielsprache befindet. Selinker (Corder 1981) prägte den Terminus *Interlanguage* und kontextualisierte später seine Auffassung von Lernersprache im Rahmen von Kontrastiver Analyse und Fehleranalyse (Selinker 1992). Granger argumentierte im Jahr 2002, dass eine Grundvoraussetzung für die erfolgreiche Entdeckung und Bearbeitung der Fehler in CALL-Systemen die Untersuchung der Fehler ist, die die Lernende in ihrer Muttersprache begehen.

Der Zusammenhang zwischen dem Begriff „Interlanguage“ und der formalen Grammatik (vgl. Paragraph 3.1.1.) kann Schwierigkeiten bereiten, da der Parser ausschließlich auf der Textebene – beziehungsweise passender in diesem Kontext - auf der Ebene von grammatikalischen Abweichungen arbeitet. Die Beziehung zwischen Fehlertyp und dem Entwicklungsprozess der Interimssprache ist schwierig zu bestimmen. Zusätzlich können verschiedene Prozesse der Interimssprache demselben Fehler unterliegen. Jedoch können diese Prozesse eine nützliche Erklärung der zahlreichen Stadien der Lernersprache bieten, und sie heben die Systematik der Interimssprache hervor, wie von Wittich (1967:316) nach vielen Analysen der Oberflächenstruktur erklärt: „the degree of error may be indicative of the learner's stage on the interlanguage … continuum“.

Die Erstsprache ist dennoch nicht der einzige Einflussfaktor auf die Zielsprache. Aus verschiedenen Sprachhintergründen begehen Lernende oft die gleichen Fehler, die vergleichbar sind mit den Fehlern, die von Muttersprachlern gemacht werden. Brown (1980) zeigt deshalb, dass sich die Inkorrektheiten aus *interlingualem* oder *intralingualem* Transfer ergeben. Die erste Phase des Zweitsprachlernens ist von interlingualem Transfer aus der Muttersprache gekennzeichnet. Bevor ein Lernender in ein L2-System hinein wächst, ist die Muttersprache das einzige linguistische System, worauf er sich beziehen kann (*interlingual error*). Richards (1974) betrachtet die intralingualen Fehler als einen negativen Transfer der linguistischen Strukturen innerhalb der Zielsprache. Der Autor (1971) erwähnt vier Haupttypen oder Ursachen der intralingualen (entwicklungsgemäßen) Fehler:

a. Übergeneralisierung: Prozess, in dem ein Lernender eine Grammatikregel jenseits des normalen akzeptierten Nutzen verwendet;

b. Unvollständige Anwendung der Regeln: falsche Anwendung von Regeln in

einem Kontext, in dem sie nicht benutzt werden sollten; dies verursacht systematische Fehler;

c. Unwissenheit einer Regelbedingung: Die Lernenden kennen eine Einschränkung und/oder eine Ausnahme einer Regel nicht, weshalb sie einen Fehler begehen;

d. Falsch vorausgesetzter Begriff: Die Lernenden verstehen eine grundsätzliche Unterscheidung in der Zielsprache nicht (Ellis 1994). Dies nennt Keshavarz (2011) eine „falsche Analogie".

In den Studien, in denen die Fehler nach der Form (Punkt 3) eingeordnet und behandelt werden, kommt die Häufigkeit dieser grammatikalischen Fehler immer in Betracht. Wagner et al. (2007) weisen beispielsweise mittels parserbasierter Systeme nach, dass im Englischen die häufigsten Fehler die Folgenden sind: die Auslassung eines Worts, die Einfügung eines zusätzlichen Worts, die Rechtschreibung und die Kongruenz.

Für das Deutsch hat Heift (2001) eine Klassifizierung abgefasst: Sie betrachtet 1387 Fehler und beobachtet diese Verteilung auf den verschiedenen Sprachebenen: 29,5% Inkorrektheiten in der Rechtschreibung und 70,5% grammatikalische Fehler; wobei sich die grammatikalischen Fehler wiederum in 28,5% beim direkten Objekt, 18,5% in der Subjekt-Verb-Kongruenz und 17,5% in der Präpositionalphrase[33] aufteilen.

In der computergestützten Lernhilfe ist eine angemessene Fehlerklassifizierung eine Voraussetzung für eine passende Erläuterung der Fehler und die Generierung eines hilfreichen Feedbacks. Im Bereich von NLP in CALL benutzen die Kategorisierung jedoch die oberflächlichen Eigenschaften der geschriebenen Sprache; sowohl die morpho-syntaktischen als auch die lexikalischen Phänomene werden betont. Diese Klassifizierungen spiegeln oft die benutzten Grammatikformalismen und somit die Kodierung der grammatikalischen Informationen in einem NLP-System wider.

Heift und Schulze (2007) schlagen eine Klassifizierung der Fehler gemäß Taylor (1998) und Lee (2003) vor, die beide für die computergestützte Analyse der

[33]Die komplette Darstellung der Verteilung der Fehler: Heift 2001; Heift, Schulze. 2007.

Textproduktionen entwickelt wurden. Die für die Einordnung betrachteten Eigenschaften werden entlang verschiedener Achsen organisiert.

Taylor (1998) empfiehlt drei Achsen:

a. den Prozess der Interimssprache (typische Fehler: Vereinfachung, Generalisierung, Versetzung);

b. die Systematik[34];

c. die Oberfläche (auf den POS-Taggern basierende typische Fehler: Auslassung, Ersatz, Erfindung).

Die Unterscheidung von Lee (2003) setzt die Arbeit von Taylor fort und wird in der Analyse deutscher Texte verwendet. Da sich ihre Analyse auf Wörter konzentriert, sind die prinzipiellen Klassen die Folgenden: a. die oberflächliche Strategie (Auslassung, Ersatz, falsche Auswahl, falsche Reihenfolge und Missbildung); b. die linguistische Kategorie bzw. lexikalische Kategorie (POS) und syntaktische Kategorie.

Die Fehlerklassifizierung von Heift und Schulze (2007) schließt zwei Achsen ein: erstens, die oberflächliche Strategie, die Fehler beinhaltet, die von verschiedenen Bedingungsverletzungen verursacht werden (Ersatz, Missbildung, Auslassung, falsche Auswahl, falsche Reihenfolge); zweitens, *Part Of Speech* bzw. Wortklassen wie Adjektiv, Determinator, Nomen, Präposition oder Verb[35], die nicht vernachlässigt werden können (typische Fehler: schlecht geformte Morphologie, Zuordnung des Kasus, Kongruenz).

4.1.1. Die Fehleranalyse

Die Relevanz und Wichtigkeit dieses Themas wurde erst spät eingesehen, obzwar der Fehler beim Erst- und Zweitspracherwerb so alt wie der Spracherwerb selbst ist. Keshavarz (2011:53) gibt zu: „it is widely believed that language learning, like acquiring virtually any other human learning, involves the making of errors".

[34]Der Begriff basiert auf der Unterscheidung von Corder (1974): *pre-systematisch* sind die Fehler, die ein Lernende begeht, bevor er sich einer bestimmten Regel der Zielsprachgrammatik bewusst ist; in der *systematischen* Phase verlässt der Lernende sich auf eine Regel der Interimssprache; die Abweichungen in der *post-systematischen* Phase können als Lapsus betrachten werden, da der Lernende beständig die Regel der Zielsprachgrammatik benutzt, aber in diesem besonderen Fall scheitert er wegen mehrerer Faktoren.

[35]Adverbien, Partikeln, Zahlwörter werden nicht in Erwägung gezogen, da sie wenig anfällig für Fehler sind.

Während die Fehleranalyse in der Sprachwissenschaft allgemein bereits thematisiert wird (Menyuk 1971, Dulay et al. 1982; Keshavarz 2011), existiert noch keine eindeutige Theorie der Fehleranalyse oder -klassifizierung in CALL.

Am Anfang wurde das Thema der Fehler von der Schule der Kontrastiven Analyse (*Contrastive Analysis*, *CA*) (Lado 1957) als Einblick in die Muttersprache des Lernenden betrachtet. Dies wurde auch Transfer oder Interferenz genannt. Laut dieser Schule können die Sprachfähigkeiten der Lernenden einer neuen Sprache aufgebaut bzw. verbessert werden, wenn man die Sprachsysteme der L1 und der L2 miteinander vergleicht. Wenn verbundene linguistische Phänomene zwischen L1 und L2 stark abweichen (z.B. die Deklination der Adjektive), erweisen sie sich als schwierig zu erlernen und werden so zu Ursachen von Fehlern. Wenn sich diese Phänomene jedoch sehr ähneln, werden sie einfacher erworben. Gegenargumente für diese zwei Hypothesen wurden geäußert, da viele Forscher nicht die Textproduktionen der Lernenden in Erwägung zogen und prüften, sondern abstrakte Konstrukte benutzt wurden.

Die *Error Analysis* (Fehleranalyse, *EA*) (Corder 1974; Dulay et al. 1974) versucht einige Schwierigkeiten dieser vorherigen Schule zu lösen. Die Fehler sollen durch zwei wichtige Schritte behandelt werden: Zuerst soll der Fehler analysiert werden, bevor ein Vergleich mit der Muttersprache gemacht werden sollte (Gass et al. 2001). Die *Error Analysis* fokussiert die Sprache der Lernenden: In einem Text werden die Fehler als Hinweis des Lernprozesses betrachtet und stellen eine Abweichung von der Regel dar (Ellis 1994).

Laut Corder (1974) kann der Begriff „fehlerhaft“ auf verschiedene Weise verstanden werden: Einerseits kann es bedeuten, dass ein Wort oder eine Struktur rein oberflächlich von der Zielsprache abweicht, anderseits kann es aber auch bedeuten, dass ein Wort oder eine Formulierung gänzlich unpassend beziehungsweise ungrammatisch – und somit wirklich falsch – ist. Die von Corder geschilderten drei Phasen der traditionellen Fehleranalyse – Fehlererkennung, -beschreibung und -erklärung – sollen durch NLP-Techniken implementiert und verbessert werden (Menzel et al. 1999), um ein hochwertigeres Ergebnis zu bekommen. Jedoch hält ein parserbasiertes CALL-Programm die Sprachkenntnisse der Parsergrammatik und einige Phänomendarstellungen in Grenzen. Die ersten zwei Phasen können durch einen Parser mithilfe einer Feedback-Komponente geleistet werden; die dritte Phase

ist allerdings für einen morpho-syntaktisch-basierten Parser schwierig durchzuführen. Ein morpho-syntaktischer Parser beinhaltet nicht genug Informationen über andere linguistische Ebenen (*linguistic levels*) wie den situationsabhängigen Kontext und das Weltwissen im Allgemeinen. Um eine plausibel wohl geformte Äußerung zu realisieren, werden diese Informationen jedoch benötigt. Der Parser kann den schlecht geformten Satz in Beziehung zur wohl geformten Entsprechung bringen - aufgrund der verletzten Bedingung; dem Fehler liegt diese verletzte Bedingung zugrunde.

Aufgrund von zwei Argumenten wurde die Fehleranalyse dennoch kritisiert: wegen ihrer Schwachheit bezüglich der methodologischen Prozeduren – wie beispielsweise der verwendeten Fehlerklassifizierung - und wegen ihrer Einschränkungen des Ziels, d.h. ihrer Fokussierung auf die *negativen* Sprachphänomene des Lernenden (Ellis 1994).

Die parserbasierte Analyse übernimmt einige Einschränkungen des Ansatzes der Fehleranalyse. Die Lernenden bekommen normalerweise Informationen über die gemachten Fehler, und auf Grund der gefundenen Fehler erstellt der Parser ein Lernerprofil (*student profile*). Jedoch fällt der Parser im Auffinden der Ursache von Fehlern aus (Gass et al. 2001), da er nur grammatikalische Fehler betrachtet, während typische Fehler, die beispielsweise im Bereich der Semantik oder Pragmatik gemacht werden, oft übersehen werden.

4.2. Das Feedback

Nachdem die Fehlererkennung und -analyse beschrieben wurden, muss nun die Thematik des Feedbacks berücksichtigt werden. Die Rolle und der Beitrag des verbessernden Feedbacks für das Sprachlernen waren und sind zurzeit von großer Bedeutung in den Theorien von CALL und SFLLF, als auch in der Pädagogik. Es wird für gewöhnlich als technischer Ausdruck verstanden, wie Upshall (1993:337) beschreibt[36]:

> Feedback: a principle used in self-regulation control systems. Information about what is happening in a system (such as level of temperature, engine speed or size of a workpiece) is fed back to a controlling device, which compares it with what should be happening. If the two are different, the device takes

[36]Diese Erklärung muss in Verbindung mit der Vorstellung von „servo system" (Upshall 1993:892) gesetzt werden. (Schulze 2003)

> suitable action (such as switching on a heater, allowing more steam to the engine, or resetting the tools).

In der Psychologie und der Pädagogik wird der Terminus „Feedback“ häufig auf dieselbe Weise verwendet. Hierzu äußerte sich Annett (1969:26) wie folgt:

> „the concept of feedback can be applied to the analysis of behavior ranging from the simplest of movements to complex problem-solving tasks. (…) The general term used since about the turn of the century for a variety of forms of psychological feedback is knowledge of results. “

Zusätzlich unterscheidet der Autor zwischen zwei Arten von Feedback: inneres (*intrinsic*) und äußeres (*extrinsic*) Feedback. Hierbei bezieht sich das innere Feedback auf die Aufgabe (*task*) selbst. Das äußere Feedback braucht hingegen einen zusätzlichen Feedback-Zyklus (*loop*). Dies bedeutet, dass für das äußere (*extrinsic*) Feedback zusätzliche Informationen von außen – d.h. unabhängig und außerhalb der Aufgabe – gegeben werden müssen, wie bspw. Information zu dem Maßstab, der für die Leistung (*performance*) angesetzt wird (Annett 1969:27).

Den Ablauf kann man sich hierbei wie folgt vorstellen: Nachdem der Lernende dem Computer eine Eingabekette eingegeben hat, die er basierend auf seiner grammatikalischen Kompetenz verfasst hat, vergleicht der Parser deren Struktur mit jenen, die er durch die ihm zur Verfügung stehende computationelle Grammatik analysieren kann. Daraufhin generiert der Parser dem Lernenden eine Rückmeldung. Somit gleicht die Arbeit des Parsers der Leistung eines Lehrers: Anstatt jedoch auf persönliche grammatische Kenntnisse zurück zu greifen, nutzt der Parser die formale Grammatik, um ein Feedback zu geben. Verschiedene Faktoren können jedoch die Qualität des Feedbacks beeinflussen, weshalb der zugrundeliegende Ansatz, der den Parser und sein Feedback beschreibt und implementiert, vielfältig und umfangreich sein sollte. Dazu müssen im Bereich von ICALL vier Hauptaspekte im Hinblick auf Feedback beleuchtet werden: Die Psychologie und Lehrtheorie; die Fremdsprachenlehr- und Fremdsprachenlernforschung; die Interaktion zwischen Mensch und Computer, und zuletzt die Lernfähigkeit der Lernenden.

4.2.1. Die Psychologie und das Feedback

Dieser Faktor beschäftigt sich zunächst mit dem Begriff des *reinforcement*

(Verstärkung) des Behaviorismus: Beim Lernen und Erlernen allgemein wirken können Feedback als eine Art Verstärker (*reinforcer)* wirken. Da jedoch ein Großteil der Forschung aus dem Bereich der Lernpsychologie an Tieren durchgeführt wurde, bleibt die Frage, ob das Prinzip der Verstärkung (*reinforcement*) auch auf Menschen und deren Lernverhalten übertragbar ist. In diesem Rahmen argumentiert Liebermann (1990), dass das Prinzip der Verstärkung beim Menschen nur eintritt, wenn gewisse Grundsätze gegeben sind. Zu diesen zählt er unter anderem die Art und Weise der Verstärkung, deren Timing, die Motivation, sowie der Kontext bzw. das Szenario und die Art des Verstärkers (*reinforcer*). Im Bereich des computergestützten Sprachlernens kann die Verstärkung erst dann den Lernprozess beeinflussen, wenn bei der Erstellung der Übungen wiederkehrende Strukturen und Muster verwendet werden.

Skinner bekräftigt die Wirkungen einer Antwort auf die Antwort selbst und folgert, dass die Einflüsse der Verstärkung und Bestrafung unsymmetrisch sind, da die Bestrafung schwerlich eine positive Umstellung in der Antwort verursachen kann.

Die Theorie des Kognitivismus spricht hingegen dem Prinzip der Verstärkung (*reinforcement*) keinerlei Bedeutung und Einfluss beim Lernen zu. Stattdessen sieht der Kognitivismus die Verstärkung darin, dass sie eine Umsetzung von gelernten Aktionen in das Verhalten ermöglicht. Somit siedelt der Kognitivismus das Prinzip der Verstärkung nicht im Bereich des kognitiven Lernens, sondern im Bereich des aktiven Handelns und Verhaltens an.

In CALL ist das Feedback eine einflussreiche Verstärkung, wobei man jedoch darauf achten sollte, dieses nicht zu überschwänglich oder übertrieben zu formulieren, um die Macht des Feedbacks nicht zu beeinträchtigen. Wie bereits zuvor erwähnt, kann eine Form der Bestrafung (*punishment*) Nachteile mit sich bringen und bei den Lernenden Besorgnis oder Angst hervorrufen, die letztlich die Aufmerksamkeit, Motivation und Aufnahmefähigkeit hemmen. Somit sollte verhindert werden, dass eine Aktion des Computers beim Lernenden nicht als Bestrafung interpretiert wird, und Rückmeldungen und Feedback sorgfältig und bedacht formuliert werden.

Innerhalb der Psychologie des *Information-Processing* wird Feedback als eine Art Servo-Mechanismus beschrieben: Bei einer Aktion wird zwischen Input und Output ein Zyklus gebildet (*loop*), wobei der Output Informationen über die Angemessenheit des Inputs liefert. Dieser Ansatz ist dem Kognitivismus zuzuordnen. Ebenso die Gestalttheorie, die auch Tolmann (1932) annimmt: Er argumentiert, dass Lernen darin

besteht, den Ursache-Wirkungs-Zusammenhang zu entdecken und zu verstehen. Die Idee der Verstärkung (*reinforcement*) lehnt er allerdings ab - trotz der Bedeutungsähnlichkeit zu seinem Begriff der „Bestätigung“. So erklären auch Hergenhahn et al. (1997:302):

> „[d]uring the development of a cognitive map, expectations are utilized by the organisms. (…) Early tentative expectations are called hypotheses, and they are either confirmed by experience or not. Hypotheses that are confirmed are retained, and those that are not are abandoned.“

Hergenhahn et al. (1997:334) besprechen in ihrer Arbeit den Begriff der Bestärkung in der kognitivistischen Psychologie und führen auch den theoretischen Ansatz von Bandura ein, in dem die Bestärkung zwei fundamentale Funktionen hat: „First it creates an expectation in observers (see observational learning) that if they act like a model who has been seen being reinforced for certain activities, they will be reinforced also. Second, it acts as an incentive for translating learning into performance. “

Innerhalb des computergestützten Lernens spielt das Prinzip der Verstärkung (*reinforcement*) eine wichtige Rolle, da bereits bestehende CALL-Programme bewusst Wiederholungsaufgaben verwenden, und somit die Verstärkung bestimmter linguistischer Eigenschaften voraussetzen (Schulze 2003).

4.2.2. Die Fremdsprachenlehr- und -lernforschung und das Feedback

Der Begriff von *reinforcement* wurde unter den Sprachwissenschaftlern stark kritisiert: Die Termini „Reiz“ und „Reaktion“ könnten nicht in Bezug auf das Sprachlernen betrachten werden.

Studien über den Erstspracherwerb zeigten, dass die Eltern die linguistischen Fehler der Kinder selten korrigieren - somit wurde die Wichtigkeit der Bestärkung in der Sprachdidaktik angezweifelt. Ellis (1994:300) erläutert die Natürlichkeit des Spracherwerbs eines Menschen mit dem Ausdruck „developmental in nature“. Dennoch ist Chomskys Hypothese der Universalen Grammatik für die Fremdsprachenlehr- und -lernforschung noch kontrovers (Cook et al. 1996) - trotz der Einstimmigkeit in der Existenz eines dem Sprachlernen gewidmeten kognitiven Apparats.

Auch die Hypothesen von Krashen (1985) wirken noch auf die gegenwärtigen Theorien der Fremdsprachenlehr- und -lernforschung (FSLLF) ein: Ihm zufolge sind

weitere kognitive Fähigkeiten neben dem Sprachlernapparat von Belang. Der Autor nimmt an, dass die Lernenden zwei Systeme für die L2-Verarbeitung haben: ein erworbenes System als Ergebnis der Kommunikation und ein gelerntes System als Resultat der bewussten Aufmerksamkeit für die Sprache und ihre Grammatik.
Somit ist deutlich, dass unabhängig von der Existenz der Universalen Grammatik und des kognitiven Apparats, viele kognitive Prozesse zum Fremdspracherwerb und -lernen beitragen und diese von Feedback und *reinforcement* profitieren können. Hierzu gibt Ellis (1994:583-584) an:

> „A number of terms have been used to refer to the general area of error treatment; these are 'feedback', 'repair', and 'correction'. 'Feedback' serves as general cover term for the information provided by listeners on the reception and comprehension of messages. As Vigil and Oller (1976) have pointed out, it is useful to distinguish 'cognitive' and 'affective' feedback; the former relates to actual understanding while the latter concerns the motivational support that interlocutors provide each other with during an interaction. 'Repair' is a somewhat narrower term used by ethnomethodologists such as Schegloff, Jefferson, and Sacks (1977) to refer to attempts to identify and remedy communication problems, including those that derive from linguistic errors. 'Correction' has a narrower meaning still, referring to attempts to deal specifically with linguistic errors; it continues an attempt to supply 'negative evidence' (…) in the form of feedback that draws the learners' attention to the errors they have made.“

Nach Ellis (1994) bewirkt das kognitive positive Feedback die Festigung der Informationen, während das negative Feedback eine von Lernenden gemachte Bearbeitung der Fehler ermöglichen sollte.
Eine ähnliche Folgerung wurde von Lyster et al. (1997) gezogen: Die Lernenden formulieren auf Anfrage einer Erklärung ihre Äußerungen um, aber nicht immer nach einer Bestätigung oder Wiederholung. Da der Lernende nach Erklärung den Fehler zu verarbeiten neigt, kann ein Computerprogramm dadurch ein positives Ergebnis bekommen - wenn es durch eine „trigger-message“ den Fehler zeigt. Wenige Beweise haben die Effizienz des negativen Feedbacks bekräftigt, jedoch bestätigt Long (1996) die Erleichterung des Sprachlernens für das Lexikon, Morphologie und Syntax.

Anhand dieser Forschungen betonen viele CALL-Programme die Notwendigkeit einer Rückmeldung für den Lernenden, um so eine Selbst-Korrektur des Lernenden zu ermöglichen

4.2.3. Die Interaktion zwischen Mensch und Computer und das Feedback

Das Fach der Interaktion zwischen Menschen und Computer beschäftigt sich mit den Themen der Schnittstellen und des Gesprächsdesigns.

Eine Unterscheidung zwischen Mensch und Maschine muss berücksichtigt und betrachtet werden, um die Interaktion des Lernenden mit dem computergestützten Programm besser zu verstehen. Erstens, weil Maschinen aus einzelnen Teilen für zielgerichtete Aufgaben erstellt werden, und Menschen dagegen holistische Entitäten sind, deren Aufgaben unterschieden werden können und müssen. Zudem verarbeiten Menschen allerhand Erfahrungen und erschaffen ihre eigene Umwelt mehrmals – eine Eigenschaft, die Maschinen nicht vorweisen können. Diese „berechnen" ein Problem basierend auf *pre-wired* bzw. vordefinierten Regeln. Außerdem bezieht sich eine Haupteigenschaft des menschlichen Denkens auf Widersprüche und die Fähigkeit, diese zu lösen: Dies kann wegen Komplexität, Varietät und Detaillierung nicht berechenbar sein (Schmitz 1992). Diese Beschreibung von Schmitz untertreibt den Wert und die Fähigkeiten eines Computers, da sie alle Operationen erfüllen können, ohne das Ziel zu erkennen.

In einer theoretischen Beschreibung der Interaktion Mensch-Computer ist der Unterschied zwischen Aktionen und Operationen in der *Activity Theory* von größter Wichtigkeit. Die Befürworter dieser Theorie betrachten die Menschen als soziales Wesen, deren Lebensweise in Tätigkeiten bzw. sozialen Interaktionen und in Aktionen innerhalb einer Umwelt besteht. Nach Leontiev (1981) kann die Aktivität mit einer Motivation, und die Aktion mit einem Zweck verbunden sein, und die Operation korreliert mit den Bedingungen. Die kommunikativen Aktivitäten können in Aktionen eingeteilt und die Aktionen wiederum in Operationen unterteilt werden. Die Operationen werden dann normalerweise als Aktionen gelernt. Die Menschen lernen somit schwierige Tätigkeiten auszuführen, wenn sie bestimmte Operationen in einer bestimmten Reihenfolge durchzuführen lernen. Andererseits werden die Maschinen nur dazu gestaltet, um gewisse (oft komplizierte) Operationen auszuüben und jede Maschine wird von einer Bedingung – z.B. einem Mausklick, einer Eingabetastatur – initialisiert. Da sie keinem Ziel unterliegen, können sie nicht als Aktionen betrachten werden (Schmitz 1992:169).

Die *Activity Theory* ist für das Verständnis der Interaktion zwischen dem Lernenden und dem Sprachlehrprogramm erheblich. Der Lernende deutet die Operationssequenz

des Computers als eine Aktion und reagiert dementsprechend. Dies erklärt, weshalb viele Lernende sowohl mit einem Computer als auch mit einem menschlichen Tutor unzufrieden – gar frustriert – sein können, wenn eine Antwort fehlerhaft ist und abgelehnt wird. Diese Fallgruben sollten von einem idealen computergestützten System vermieden werden.

In der Interaktion zwischen Mensch und Computer, besonders in ICALL- und ILTS-Programmen, muss ein Dialogsystem implementiert werden. Sutcliffe (1995) listet die wichtigsten Prinzipien eines Dialogsystems auf und meint mit dem Terminus „Feedback“ die fortlaufend dem User gegebenen Information über den Verlauf. Mit Rücksicht auf die Fehlermitteilungen werden vier Prinzipien umrissen (Schneiderman 1992): die Genauigkeit bzw. Spezifizität, die konstruktive Ableitung und positiver Ton, auf den Benutzer zugeschnittene Formulierungen und angemessenes technisches Format.

Darüber hinaus bespricht Schneiderman das Timing einer Rückmeldung: Ein Feedback in kurzer Zeit sei schnell und angenehm; umgekehrt rege ein länger dauerndes Feedback den User dazu an, die Aufgaben sorgfältiger auszuüben. Kulik et al. (1988) folgern, dass die Effizienz einer Rückmeldung auf die Geschwindigkeit ankommt: je unmittelbarer das Feedback erfolgt, desto effektiver ist es.

4.2.4. Die Lerner-Reaktion auf Feedback in CALL

Ein parserbasiertes Programm zielt auf die Zuweisung von Informationen über den Fehler ab. Ein Lernender muss mit einer aussagekräftigen und gehaltvollen Analyse seines/ihres Inputs versorgt werden. Die Fehleranalyse des computergestützten Systems bildet dabei die Basis eines Feedbacks.

Laut Garrett (1987) bestehen vier grundlegende Sorten von Feedback in CALL:

1. Das System zeigt nur die richtige Antwort auf;
2. Das System lokalisiert einen Fehler durch einen Vergleichung zwischen der Eingabekette des Lernenden und der in der Datenbank gespeicherten richtigen Darstellung;
3. Die Rückmeldungen werden mit möglichen Fehler verbunden und im System gespeichert und werden gezeigt, wenn die Antwort des Lernenden zu den Fehlern passt;

4. Das System verwendet einen intelligenten NLP-Ansatz – bzw. einen Parser -, mit dem der Computer eine Sprachanalyse der Antwort des Lernenden vollziehen kann.

Das Feedback muss einige besondere Eigenschaften haben; van der Linden (1993) empfiehlt Genauigkeit, Schlichtheit und Prägnanz: Beispielsweise wird eine Rückmeldung, die sich über mehr als drei Zeilen erstreckt, nicht gelesen und scheint somit nutzlos.
Trotz vieler Studien über die beste Qualität des Feedbacks wird die Rückmeldung des Formats „richtig/falsch“ in CALL-Programmen am meisten benutzt (Kreindler 1998).

Man betrachte als Beispiel den *German Tutor* (Vorgänger von *E-Tutor*: vgl. Abschnitt 2.2.2.): Die Autoren des Programms wollten sich auf die hochfrequenten Fehler konzentrieren sowie auf die hilfreiche Rückmeldung, die informativ und fehlerbasiert sein sollte (Heift et al. 2001).
Mit einer Studie beweist Heift (2001), dass 40% der analysierten Sätze schätzungsweise mehr als einen einzigen Fehler beinhalten. Bei einem pädagogischen und motivierenden Standpunkt muss ein System lehrreiches Feedback geben. Heift argumentiert hierzu, dass es lernerfreundlicher ist, bei Sätzen mit mehreren Fehlern ein Feedback pro Fehler zu geben. Ein Feedback, das alle Fehler eines Satzes enthält ist nicht förderlich für die Lernende. Bezieht man jedoch einen computationellen Standpunkt, so treten einige Fragen auf: Welcher Ordnung müssen die Fehler folgen? Wie kann diese Kenntnis über Parser und deren Feedback in ein CALL-System integriert werden?
Heift et al. (2000a) wählen eine „Error Priority Queue“ aus, bzw. ein System, das die Fehler einschätzt und eine einzige Rückmeldung zeigt.

Laut Nagata et al. (1995) können der Computer und generell computergestützte Systeme die grammatikalische Bewusstheit steigern, wenn sie dem Lernenden ein fehlerbasiertes Feedback anbieten. Die Sprachfertigkeiten und das Alter eines Lernenden beeinflussen die Präferenz für einen bestimmten Typ von Rückmeldung in einem ICALL-System. Außerdem reagieren die Lernenden unterschiedlich auf das Feedback: Sie können die Fehler verbessern; im Fall von mehrfachen Fehlern korrigieren sie nicht die Fehler, die das System nicht rückgemeldet hat; sie können

eine richtige Struktur ändern; sie legen die gleiche Äußerung wieder vor; sie fragen nach der korrekten Lösung (Heift 2001). Wenn jedoch kein Feedback zu dem Fehler erzeugt wird, bekommen die Lernenden nicht die Möglichkeit, diesen zu korrigieren. Es folgt daraus, dass in einem ICALL-System die Lernenden das Feedback erwarten, um die Fehler selbstständig berichtigen zu können.

Die Erklärung im Feedback sollte deshalb viele verschiedene Variablen erwägen. Chapelle (1998:23) listet sieben Hypothesen „relevant for developing mutimedia CALL" auf, darunter auch die Wichtigkeit der Beachtung eigener Fehler im Output und in der Selbst-Korrektur. Diese Berücksichtigungen können dem Lernenden beim Verinnerlichen der grammatikalischen Form helfen.

Nicht nur die Qualität der Interaktion zwischen Computer und Lernenden, sondern auch die Lernfähigkeit eines Lernenden hängt vom Typ des Feedbacks ab. Mit Rücksicht auf SFLLF-Theorien werden die folgenden Rückmeldungen von einem Lehrenden am meistens benutzt (Lyster et al. 1997; Ferreira et al. 2005): explizite Korrektur; Umschreibung (*recast*); Klärung; meta-linguistisches Feedback[37]; die Aufforderung an die Lernenden, den Fehler selbst zu verbessern – ohne eine genaue Fehlererklärung durch das System (*elicitation*); Wiederholung.

In einem CALL-Bereich können weder die explizite Korrektur noch die Umschreibung eine Verbesserung in der Sprache des Lernenden hervorrufen. Die Klärung bietet keine Hilfe in der Selbst-Korrektur der Fehler, aber sowohl *elicitation* als auch die Wiederholung können den Fehler betonen. Heift (2004) beweist, dass das meta-linguistische Feedback sinnvoll, hilfreich und nützlich ist, besonders wenn der Fehler im Output beleuchtet wird.

4.3. Die Fehlertypologie im Trainingskorpus

Wie Wagner et al. (2007) bestätigen, erfordert die Kreation eines fehlerhaften Korpus – d.h. mit ungrammatikalischen Sätzen – viel Zeit und eine tiefe Kenntnis der betrachteten Sprache. Die Sprachkenntnisse (*linguistic knowledge*) in GI-Tutor bestehen aus einer Datenbank von fehlerhaften und entsprechend richtigen Sätzen, die das verwendete Korpus darstellen.

[37]Der Lernende bekommt eine sorgfältige Erklärung des Fehlers.

Ein relativ kleines Fehlerkorpus wurde in einer vergangenen Studie (Terzarede 1999/2000) über die häufigsten Fehler, die italienische Muttersprachler beim Lernen des Deutschen machen, gesammelt.
Im Rahmen der vorliegenden Arbeit wurde dieses Korpus weiterentwickelt: Textproduktionen, die 2014 als Klausuren des Kurses „Deutsch 1“ an der Universität Ca' Foscari geschrieben wurden, wurden unter Wahrung der Anonymität untersucht und fehlerhafte Sätze daraus wurden digitalisiert.
Insgesamt umfasst das Korpus 1252 Sätze: 626 fehlerhafte Sätze und 626 entsprechend korrigierte Sätze. Die fehlerhaften Sätze können sowohl einen als auch mehrere Fehler beinhalten.

Dieses Korpus möchte ein Gesamtbild der typischen Fehler darstellen, die italienische Muttersprachler im Deutschen begehen. Jedoch ist der Begriff von Fehler sehr „subjektiv“ (Lennon 1991): Beispielsweise kann ein Muttersprachler des Deutschen eine gegebene Äußerung als akzeptabel erachten, auch wenn diese Äußerung nicht ganz verständlich ist. Diesbezüglich besteht Chomksy (1965) auf der Grenze zwischen Akzeptabilität und Unakzeptabilität: Akzeptabel bedeutet nicht-grammatikalisch, da sich die Akzeptabilität auf die Leistung (*performance*) der Lernenden bezieht, während die Grammatikalität auf der Kompetenz (*competence*) beruht.
Die gesammelten Sätze wurden nicht von deutschen Muttersprachlern geschrieben, somit können sie sowohl wegen der Schreibweise als auch wegen des Inhalts ungewöhnlich beurteilt werden. Zudem sind manche Fehler kontextabhängig. Außerdem muss berücksichtigt werden, dass diese Sätze von Schülern und Studenten formuliert wurden, die einen eher begrenzten Wortschatz und erst geringe Sprachkenntnisse des Deutschen haben.

Trotz der begrenzten Datenmenge wurde eine kleine statistische Analyse der Fehlertypen durchgeführt, um die wichtigsten Problembereiche der Lernenden festzustellen. Hierzu betrachte man die nachfolgende Tabelle (Tabelle 1):

Fehlertyp	Prozentsatz
Syntaktische Fehler	37,22%
Morphologische Fehler	36,10%
Lexikalische Fehler	14,70%
Orthographische Fehler	11,98%

Tabelle 1: Prozentsätze der Fehler.

Der größte Anteil der Fehler kann somit der Ebene der Syntax zugeordnet werden.
In den nachfolgenden Abschnitten werden die verschiedenen Fehlergruppen dargestellt.

4.3.1. Zuordnung der Fehler

In dieser Arbeit werden die Fehlerkategorisierungen sowohl von Terzaredes als auch von Marino nicht übernommen, da die darin enthaltenden Übersichtstabellen über die typischen Fehlerkategorien eher undeutlich dargestellt sind.

Die Fehler werden somit nach den Hauptsprachebenen gegliedert. Daraus ergeben sich vier Fehlergruppen, die nach spezifischen Eigenschaften in Untergruppen weiter eingeteilt werden. Aus diesem Grunde werden auch jene Sätze, die mehr als einen Fehler enthalten, wiederholt, um jeden dieser Fehler der richtigen Gruppe zuordnen zu können.

Die Fehlergruppen des Korpus beinhalten sowohl die fehlerhaften Sätze, die in Klammern gesetzt wurden, als auch die korrigierten Sätze. Die Beispielsätze werden hier vorgeführt, wie sie in GI-Tutor digitalisiert und verarbeitet werden. Dabei wurden die Sätze in Klammern gesetzt, damit der Parser feststellen kann, ob eine Eingabekette (*Input String*) als Äußerung erkannt wird; das Zeichen „fp“ zeigt dem Parser, dass es das Debugging-Vorgehen behandelt; die Abkürzungen „b1“ „c5“ „h7“ bezeichnen die unterschiedlichen Fehlergruppen und die entsprechende Satznummer; nach den eckigen Klammern wird der Satz als eine Liste von durch Kommas getrennten Wörtern geschrieben.

Groß- und Kleinschreibung, Umlaut, scharfes-S (ß) und Interpunktion spielen keine Rolle in der automatischen Analyse mit GI-Tutor, weil sie in diesem

Zusammenhang nur als Regeln der Rechtschreibung und nicht einer Parsing-Strategie zu betrachten sind.

Der nun folgende Abschnitt zeigt eine Gliederung der fehlerhaften Sätze. Dabei werden für jede Fehlerkategorie drei Beispiele ausgewählt und anhand deren eine kurze Beschreibung bzw. Erklärung der Fehler gegeben. Zuletzt soll darauf hingewiesen werden, dass sich die unten gelisteten fehlerhaften Sätze auf spezifische Klausuren beziehen, und manche Sätze – aus dem Kontext der Klausuren genommen – eigentlich den Regeln der deutschen Sprache entsprechen. Jedoch waren sie im Kontext der Aufgabestellungen der Klausuren fehlerhaft und sind somit Teil hier verwendeten Korpus.

4.3.1.1. Orthographie

Die Verletzung der orthographischen Regeln zeigt keinen Mangel an der Sprachkompetenz im weiteren Sinn, denn die Beachtung dieser Regeln betrifft nur die schriftliche Kompetenz.

Diese Fehler werden größtenteils wegen Interferenz der Muttersprache oder einer anderen erlernten Sprache gemacht.

Im Korpus können 36 orthographische Fehler auf Interferenz des Englischen zurückgeführt werden. Das Englisch ist üblicherweise die erste Fremdsprache, die in der Schule gelernt wird, und die Ähnlichkeiten zum Deutschen können die Lernenden dann beim Erlernen dieser Sprache irreführen.

```
fp(b4,[sie, war, ueberrauscht, .]).
```
Sie war überrascht.

```
fp(b6,[meine, schulhe, liegt, gegenueber, meinem, haus, .]).
```
Meine Schule liegt gegenüber meinem Haus.

```
fp(b7,['Bastia', lieght, in, 'Umbrien', .]).
```
Bastia liegt in Umbrien.

4.3.1.1.1. Interferenz aus dem Englischen

```
fp(c33,[ihr, name, was, sara, .]).
```

Ihr Name war Sara.

```
fp(c2,[heisst, die, person, links, under, 'Karin', ?]).
```

Heißt die Person links unter Karin?

```
fp(c4,[mein, father, heisst, 'Franco', und, mein, brother,
'Marco', .]).
```

Mein Vater heißt Franco und mein Bruder Marco.

4.3.1.1.2. Interferenz aus dem Italienischen

```
fp(d4,[diese, abend, darf, ich, nicht, in, die, disco,
gehen, .]).
```

Diesen Abend darf ich nicht in die Disko gehen.

```
fp(d6,[ich, stelle, den, kassettenrecorder, auf, den,
balcon, .])
```

Ich stelle den Kassentenrecorder aud den Balkon.

```
fp(d10,[die, lehrerin, ist, ab, und, zu, sehr, lunatisch, .]).
```

Die Lehrerin ist ab und zu sehr launisch.

4.3.1.2. Lexikalische Fehler

Ein weiteres Problem für die Lernenden ist die Selektion der geeigneten Präpositionen, die entweder vom Verb (sie führen eine Ergänzung ein) oder von ihrer Semantik (sie leiten eine Angabe ein) abhängig sind.

Die Präpositionen können durch einen Mechanismus der Vereinfachung ausgelassen oder durch eine ähnliche Klangstruktur mit Wörtern der Muttersprache (oder einer anderen gelernte Fremdsprache) falsch gewählt werden.

Die Lernenden mit italienischer Muttersprache machen viele Fehler in der Selektion der Verbpartikel „zu“, welche Infinitivsätze kennzeichnet.

4.3.1.2.1. Selektionsfehler der Präpositionen

```
fp(e2,[er, verschwendet, das, geld, an, alkohol, .]).
```

Er verschwendet das Geld mit Alkohol.

```
fp(e3,[sie, wollte, nach, berlin, studieren, .]).
```

Sie wollte in Berlin studieren.

```
fp(e5,[sie, gingen, auf, die, bibliothek, .]).
```

Sie gingen in die Bibliothek.

4.3.1.2.1.1. Selektionsfehler der Verbpartikel „zu“

```
fp(f1,[sie, entschied, sich, nach, muenchen, fahren, .]).
```

Sie entschied sich nach München zu fahren.

```
fp(f2,[sie, ging, in, die, kantine, zu, essen, .]).
```

Sie ging in die Kantine essen.

```
fp(f4,[sie, entschied, nach, berlin, zu, zurueckfahren, .]).
```

Sie entschied nach Berlin zurückzufahren.

4.3.1.2.2. Unpassende Hilfsverben

```
fp(g1,[meine, alte, tante, hat, oft, zu, uns, gekommen, .]).
```

Meine alte Tante ist oft zu uns gekommen.

```
fp(g2,[ich, habe, abgefahren, .]).
```

Ich bin abgefahren.

```
fp(g3,[ich, habe, angekommen, .]).
```

Ich bin angekommen.

4.3.1.2.3. Semantik bzw. Unangemessenheiten semantischer Art

```
fp(h1,[wo, machen, sie, die, fremdsprachen, ?]).
```

Wo lernen sie Fremdsprachen?

```
fp(h2,[wir, machten, tennis, spielen, .]).
```
Wir mögen Tennis spielen // Wir spielten Tennis.

```
fp(h3,[mein, bruder, ist, sehr, gut, ski, .]).
```
Mein Bruder kann sehr gut Ski fahren.

4.3.1.3. Morphologie

Auf der morphologischen Sprachebene werden Kompetenzen in der Flexion als auch der Deklination – bspw. Kasusbildung – gebraucht. Zumeist werden hierbei Kongruenzfehler gemacht. Die Lernenden irren sich in der Wahl des Kasus, Genus oder des Numerus eines Terminus.

Bei der Verbbildung scheint hauptsächlich die Bildung der Vergangenheitsform problematisch zu sein: Sie neigen dazu, die Vergangenheitsbildung der schwachen Verben auf starke Verben zu übergeneralisieren.

4.3.1.3.1. Kongruenzfehler der Nomen, Pronomen, Adjektive und/oder Artikel

```
fp(i1,[nachdem, er, viele, anzeige, mit, stellenangeboten,
gelesen, hatte, .]).
```
Nachdem er viele Anzeigen mit Stellenangeboten gelesen hatte.

```
fp(i2,[er, wohnte, mit, seinem, eltern, in, dem, dorf, .]).
```
Er wohnte mit seinen Eltern in dem Dorf.

```
fp(j29,[sie, organisierten, seine, reise, .]).
```
Sie organisierten ihre Reise.

4.3.1.3.2. Kongruenzfehler der Verben

```
fp(k1,[weil, in, der, mensa, viele, leute, war, .]).
```
Weil in der Mensa viele Leute waren.

```
fp(k3,[wenn, 'Thomas', mich, anrufen, bin, ich, niemals, zu,
```

```
hause, .]).
```
Wenn Thomas mich anruft, bin ich niemals zu Hause.

```
fp(k39,[in, den, sommerferien, moechtest, ich, dich,
besuchen, .]).
```
In den Sommerferien möchte ich dich besuchen.

4.3.1.3.2.1. Konjugationsfehler der Vergangenheit

```
fp(l1,[lizzie, kommte, aus, england, .]).
```
Lizzie kam aus England.

```
fp(l2,[sie, verbracht, viele, stunden, mit, den, leuten, .]).
```
Sie verbrachte viele Stunden mit den Leuten.

```
fp(l3,[ich, nehm, eine, margherita, .]).
```
Ich nahm eine Margherita.

4.3.1.4. Syntax

In der Syntax liegt wie erwähnt der Großteil der Probleme: Sowohl das Italienische als auch das Deutsche haben eine ziemlich freie Wortstellung, wobei jedoch die grundlegende Reihenfolge der Satzteile nicht übereinstimmt. Das Italienisch ist grundlegend eine SVO-Sprache (Subjekt-Verb-Objekt), während das Deutsche eine SOV-Sprache (Subjekt-Objekt-Verb) ist. Zudem zeigt das Deutsche das finite Verb immer an zweiter Stelle (Verb-Zweit-Regel) - abgesehen von Nebensätzen, in denen der Komplementierer das Feld besetzt und hier das Verb an der letzten Stelle realisiert wird[38].

Lernende mit italienischer Muttersprache verletzen typischerweise die deutsche Syntaxstruktur, besonders die Verb-Zweit-Regel (V2-Regel).

In Bezug auf die Negation sind die grammatikalischen Regeln zwischen den zwei

[38] Die Vorstellungsregel des Deutschen Satzes verlangt die Bewegung des finiten Verbs an die zweite stelle im Satz (V2-Bewegung), wobei das Vorfeld (die erste Position vor dem Verb) von dem Subjekt oder einer Ergaenzung besetzt werden kann. Jedoch ist in Nebensaetzen das finite Verb an letzter Position, da die zweite Stelle schon belegt ist und das Verb daher nicht an diese bewegt werden kann (V2-Bewegung).

Sprachen unterschiedlich: Im Deutschen folgt die Negation den Ergänzungen und den Nominalphrasen; sie geht den Zeit-, Orts- und Modusergänzungen, den prädikativen und adjektivischen Phrasen voraus. Die italienischen Muttersprachlicher haben jedoch eine pre-verbale Negation erworben und neigen daher dazu, auch das Deutsche pre-verbal zu negieren (Lightbown, Spada 2006).

4.3.1.4.1. Verletzung der Satzstruktur

```
fp(m1,[sie, amuesierten, sich, in, der, hauptstadt, auch, .]).
```

Sie amüsierten sich auch in der Hauptstadt.

```
fp(m2,[naemlich, kam, er, aus, england, .]).
```

Er kam nämlich aus England.

```
fp(m3,[als, thomas, das, erste, mal, sah, sie, .]).
```

Als Thomas sie das erste Mal sah.

4.3.1.4.1.1. Falsche Verwendung der V2-Regel

```
fp(n2,[aus, diesem, grund, monika, half, petra, .]).
```

Aus diesem Grund half Monika Petra.

```
fp(n5,[das, naechste, jahr, die, kinder, machten, den,
urlaub, .]).
```

Das nächste Jahr machten die Kinder den Urlaub.

```
fp(n34,[im, bus, 'Roland', hat, bemerkt, dass, die,
monatskarte, zu, hause, war, .]).
```

Im Bus hat Roland bemerkt, dass die Monatskarte zu Hause war.

4.3.1.4.1.2. Stellungsfehler der notwendigen bzw. fakultativen Ergänzungen im Mittelfeld

```
fp(o1,[mein, opa, bringt, ihr, sie, .]).
```

Mein Opa bringt sie ihr.

```
fp(o2,['Fritz', leiht, ihm, ihn, .]).
```
Fritz leiht ihn ihm.

```
fp(o12,[die, heutige, jugend, ist, nun, vollkommen, von, dem,
fernsehen, abhaengig, .]).
```
Die heutige Jugend ist nun von dem Fernsehen vollkommen abhängig.

4.3.1.4.1.3. Stellungsfehler der Negationspartikel „nicht“

```
fp(p38,[sie, hatte, nicht, freizeit, .]).
```
Sie hatte keine Freizeit.

```
fp(p1,[am, abend, ist, mein, vater, sehr, muede, nicht, .]).
```
Am Abend ist mein Vater nicht sehr müde.

```
fp(p18,['Peter', ist, mein, lieblingsfreund, nicht, .]).
```
Peter ist nicht mein Lieblingsfreund.

4.3.1.4.1.4. Unpassende Behandlung der trenn- und untrennbaren Verbteile

```
fp(q30,[er, teilnahm, an, diesem, italienischen,
fernsehprogramm, .]).
```

```
fp(q31,[wir, ausgingen, am, abend, .]).
```
Wir gingen am Abend aus.

```
fp(q1,['Maria', laedt, ihre, freunde, zum, mittagessen, .]).
```
Maria lädt ihre Freunde zum Mittagessen ein.

4.4. Ergebnisse der Analyse

Alle Sätze des Trainingskorpus wurden sowohl mit ParZu als auch mit GI-Tutor analysiert, wobei beide Parser vielversprechende und positive Ergebnisse erzielten.

Eine Analyse wurde als falsch bewertet, wenn: a. Die Tags irrtümlich zugeteilt wurden (da falsche Etiketten die Struktur beeinträchtigen können); b. Die Satzstruktur teilweise oder komplett unrichtig dargestellt wurde; c. Sowohl die Tags als auch die Satzstruktur fehlerhaft waren.

Dagegen ist eine Analyse als richtig betrachtet worden, wenn: a. Alle Tags des Satzes zutreffend zugeteilt wurden; b. Sowohl die Tags als auch die Satzstruktur richtig dargestellt wurden.
Die Rückmeldungen der Systeme wurden in der Untersuchung nicht betrachtet, da ParZu keine Feedbacks erzeugt und überdies einige Feedbacks für bestimmte Fehler in GI-Tutor noch zu implementieren sind.

Beide Systeme haben die Analyse aller 1252 Sätze vollständig zu Ende geführt. Jedoch werden einige Sprachstrukturen noch kaum erkannt und deshalb oft falsch analysiert - wie beispielsweise Genetivobjekte, Relativsätze, Appositionen und Partikeln wie z.B. *zu*, *um*.

ParZu wird mit keinem Lernzweck angewandt; er kann somit nicht als Lernprogramm angesehen werden. Der Dependenz-Parser kann aber die Analyse fehlerhafter Sätze vervollständigen, zeigt jedoch die Fehler nicht an und erzeugt kein Feedback.
Das System berücksichtigt die Groß- und Kleinschreibung, um bestimmen zu können, ob ein Wort bspw. ein Adjektiv oder ein Nomen ist (z.B. französisch / Französisch). Die Stammvokale mit Umlaut können einfach durch Diphthonge dargestellt werden. Allerdings kontrolliert das System nicht die Korrektheit der Endungen, womit es Phänomene wie die Kongruenz (bspw. zwischen Artikeln und Nomen) nicht überprüfen kann.
ParZu vollzog die Analyse vorwiegend richtig, jedoch generiert der Algorithmus in Einzelfällen – z.B. in Bezug auf die Konjugationsfehler - nicht einen einzigen Dependenzbaum, sondern eine Reihe von disjunkten Denpendenzbäumen (Abb.13) - Dependenzwald genannt (*dependency forest*: Kübler et al. 2009:15). Wie Kübler (2009) vorschlägt, müsste der Algorithmus unaufwendig verändert werden, damit ein Dependenzbogen die Wurzeln jedes disjunkten Baumes verbindet, um einen einzigen Baum zu bilden.

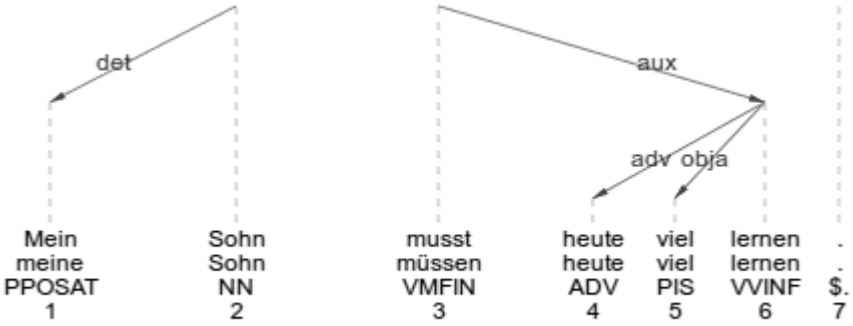

Abb. 13 : Analyse des Satzes „Mein Sohn musst heute viel lernen“.

Im Gegensatz zu ParZu, wurde GI-Tutor für ein reines Lernzweck entworfen und realisiert. Sein Zweck ist es, trotz der Fehler die Analyse bis zum Ende zu vollziehen und dabei die Fehler zu erkennen. Deswegen akzeptiert der Parser sowohl die italienischen als auch die deutschen syntaktischen Strukturen der Konstituenten.
Auch dieses System vollzog die Analyse größtenteils korrekt. Wie schon erwähnt, hat die Groß- und Kleinschreibung keine Bedeutung, jedoch beeinflussen die falsch geschriebenen Wörter, wie z.B. in der Gruppe der Orthographiefehler oder der Konjugationsfehler der Vergangenheit, das Vorgehen der Analyse. Diese Wörter werden tatsächlich mit einem unrichtigen Tag etikettiert, weil das System ein Wort sucht, das dem Inputwort eventuell noch ähnlicher sieht.

Der Verfahrenansatz wurde weitgehend auf der Robustheit bei Konjugationsfehlern – bzw. Subjekt-Verb-Kongruenz - und bei der Erkennung der Relativpronomen getestet.

```
?- parsefunct(k5).
mein sohn musst heute viel lernen .


############################################################
                TAGGING DELLA FRASE DA ANALIZZARE
############################################################

[mein-[poss],sohn-[n],musst-[vsupp],heute-[avv],viel-
[ag,avv,in,pron],lernen-[viin,vin,vit,vt],. - [punto]]

############################################################
                DISAMBIGUAZIONE DEI TAGS AMBIGUI
############################################################
```

```
[mein-[poss],sohn-[n],musst-[vsupp],heute-[avv],viel-
[ag,avv,in,pron],lernen-[viin,vin,vit,vt],. - [punto]]

##############################################################
                STRUTTURA IN COSTITUENTI SINTATTICI:
##############################################################

sogg-[sn-[poss-[mein],n-[sohn]]]
ibar-[
     vsupp-[musst]
     savv-[avv-[heute]]
          savv-[avv-[viel]]
ibar2-[
               viin-[lernen]
          punto-[.]

##############################################################
                FEEDBACK DEL CORRETTORE GRAMMATICALE:
##############################################################

Accordo Soggetto-Predicato sbagliato
musst
seconda persona singolare
soggetto =
terza persona singolare
f-[feats-   (ms3-[acc,nom]),sogg-[mein,sohn],pred-muessen,pred-
lernen]
true.
```

Bezüglich der Relativsätze kann man anführen, dass sie schon in der Psycholinguistik als schwierig zu analysieren gelten. Letztlich spiegelt sich diese Schwierigkeit in dem Prozess des Parsings wieder. Die Komplexität dieser Strukturen findet sich in der Anbindung des Relativsatzes an eine Nominalphrase - besonders wenn diese das Subjekt ist. Eine ähnliche Problematik zeigt sich bei reflexiven Verben und den Reflexivpronomina (Delmonte 2002).

Sowohl die einfachen Relativsätze, die direkt an das Nomen anschließen, als auch die extraponierten Relativsätze können von dem Parser nicht verlässlich erkannt und analysiert werden.

Die Untersuchung der Relativsätze konnte – und kann derzeit – mit GI-Tutor kaum korrekt zu Ende geführt werden. Jedoch wurden die Regeln, die das Parsing dieser Strukturen erlauben, im Programm implementiert, um bessere Ergebnisse zu bekommen.

Nachfolgend wird ein Beispiel vorgestellt, das aber die richtige Version des Satzes verwendet; das Feedback des Parsers wird in diesem Fall ausgelassen.

```
?- parsefunct(mm14).
Canetti beschreibt die grosse leidenschaft die er und seine
mutter fuer die lektuere haben .

Tags:[Canetti-[np],beschreibt-[vt],die-[art,rel],  grosse[ag],
leidenschaft-[n],  die-[art,rel],  er-[pers,pron],  und-[cong],
seine-[poss],mutter-[n], fuer-[p], die-[art,rel], lektuere-[n],
haben-[auair,ausa,ausai,vc],. - [punto]]

Konstituent-Struktur:
sogg-[sn-[np-[Canetti]]]
ibar-[
     vt-[beschreibt]
svt-[
     ogg-[sn-[art-[die],ag-[grosse],n-[leidenschaft]]]
f2-[rel-[die],    fyesno-[sogg2-[sn-[pers-[er]],    cong-[und],
snsempl-[poss-[seine],
n-[mutter]]], sp-[p-[fuer], snsempl-[art-[die], n-[lektuere]]],
ibar2-[vc-[haben]]]]
punto-[.]
```

Wie man in der Tabelle 2 entnehmen kann, hat ParZu im Allgemeinen bessere Ergebnisse erzielt.

Prozentsätze der richtigen Analyse aller Sätze		
	GI-TUTOR	**PARZU**
Orthographische Fehler	71,33%	77,33%
Lexikalische Fehler	72,83%	90,76%
Morphologische Fehler	81,19%	80,31%
Syntaktische Fehler	72,32%	79,18%
	74.41%	**81,89%**

Tabelle 2: Prozentsätze der Ergebnisse.

Die unten stehende Graphik zeigt die Resultate der vier Hauptkategorien der Fehler. GI-Tutor hat nur in der Analyse der morphologischen Fehler gering bessere Ergebnisse erzielt, während ParZu in der Analyse der orthographischen, lexikalischen und syntaktischen Fehler besser war. Diese prozentualen Ergebnisse wurden allerdings anhand aller allgemeinen Resultate errechnet. Tatsächlich bewies bei der Analyse der fehlerhaften Sätze GI-Tutor bessere Fähigkeiten (Tabelle 3).

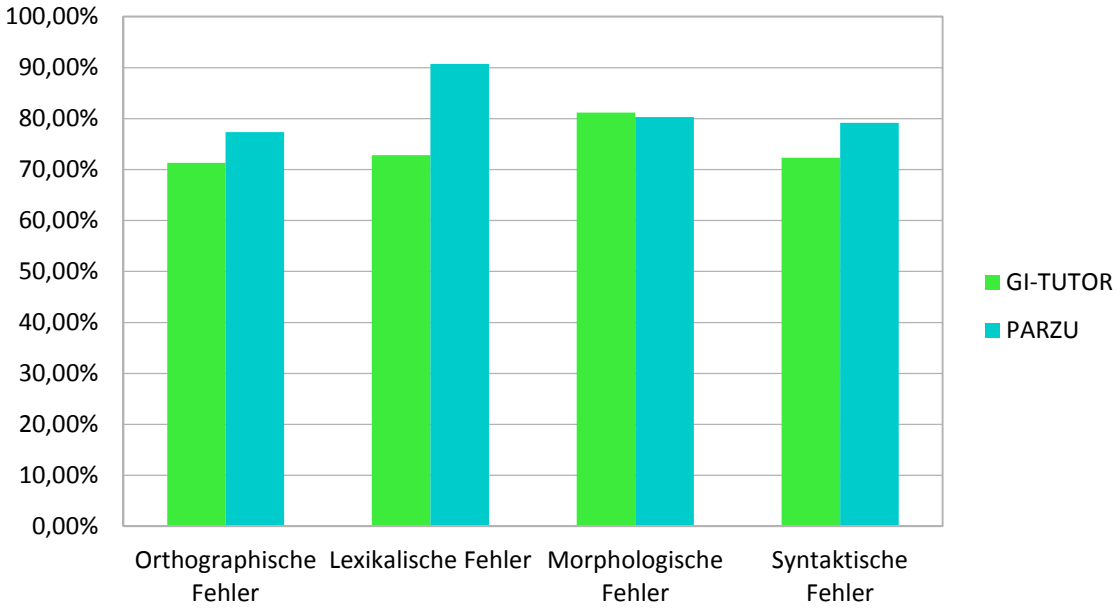

Abb. 14: Grafik 1.

		GI-Tutor		ParZu	
		Falsche Analyse	**Richitge Analyse**	**Falsche Analyse**	**Richtige Analyse**
Orthographie					
Fehlerhafte Sätze	75	30	42	27	48
Richtige Sätze	75	10	65	7	68
		28,66%	71,33%	22,66%	77,33%
Lexik					
Fehlerhafte Sätze	92	23	69	16	76
Richtige Sätze	92	27	65	1	91
		27,17%	72,82%	9,23%	90,76%
Morphologie					
Fehlerhafte Sätze	226	47	179	81	145
Richtige Sätze	226	38	188	8	218
		18,80%	81,19%	19,69%	80,31%
Syntax					
Fehlerhafte Sätze	233	69	164	79	154
Richtige Sätze	233	60	173	18	215
		27,68%	72,32%	20,81%	79,18%

Tabelle 3: Resultate.

4.5. Die Übungen von GI-Tutor

Wie schon in den vorgehenden Abschnitten erklärt wurde, ist GI-Tutor ein *shallow* Parser.

In der Benutzeroberfläche bietet GI-Tutor den Lernenden drei verschiedene Arten von Übungen, mit denen sie ihre Sprachkenntnisse testen und verbessern können.

Da die Lernenden italienische Muttersprachler sind und geringe Kenntnisse der deutschen Sprache haben, werden sowohl die Befehle als auch das Feedback auf Italienisch gegeben, damit sie die Anweisung vollständig verstehen können.

Zu Beginn muss der Lernende seinen Namen eingeben. Das System kann damit ein detailliertes Profil des Studentes anlegen. Der Lernende bekommt dann entweder mehr oder weniger Punkte für jede richtige bzw. falsche Antwort. Zuletzt kann das System nach der Übungssitzung eine Bewertung geben. Auf diese Weise kann der Lernende eine Selbstbewertung erhalten

In der ersten Übung „Scegliere la frase corretta“ muss der Lernende unter drei oder vier vorgegebenen Äußerungen den richtigen Satz aussuchen.

Um die Übung automatisch zu konstruieren, werden sowohl die fehlerhaften als auch die entsprechenden korrekten Sätze verwendet: Durch diese Methode werden drei richtige Sätze willkürlich ausgewählt und danach ein fehlerhafter Satz in die Gruppe eingefügt.

Die zweite Übung ist „Trovare l'errore o gli errori“: Der Lernende bekommt einen Satz und muss den Fehler - oder die Fehler – erkennen und wenn möglich sie korrigieren.

In der dritten Übung muss der Lernende Sätze bilden: Ihm/Ihr werden eine Reihe von Lemmata gegeben, die der Satzdatenbank entnommen werden. Ferner wird das große Lexikon benutzt, in dem die Lemmata mit morphologischen und syntaktischen Eigenschaften detailliert klassifiziert sind (vgl. Anhang 1 und Kapitel 3). Dieses Wissen liegt in der Datenbank und in der Grammatik, die im Parser enthalten sind. Insofern als der Lernende mit den Lemmata jede mögliche Äußerung bilden kann, werden die Antworten nicht durch eine einfache *pattern-matching* Operation beurteilt. Der Parser muss die Korrektheit des Satzes kontrollieren.

Nach dem Feedback, in dem die Fehler detailliert gezeigt werden, muss der Lernende die Übung wiederholen. Wenn ein Satz richtig ist, bestätigt das System die Korrektheit

und gibt dem User einen neuen Satz.

Immer wenn der Lernende eine Aufgabe abbrechen möchte, bietet das System eine Bewertung bzw. Beurteilung der ganzen Interaktion. Die Ergebnisse werden dann durch eine Grafik dargestellt, wobei die rote Farbe die Punktzahl der Fehler anzeigt und die grüne Farbe die Anzahl der richtigen Sätze.

4.5.1. Feedback in GI-Tutor

Wie schon ausführlich erläutert, ist es sehr wichtig, dass die Lernenden eine Rückmeldung und, wenn nötig, eine Beschreibung der Fehler bekommen.
Die Lernenden können damit ihre sprachlichen Schwächen erkennen und weitere zweckmäßige Aufgaben zu lösen versuchen, um ihr Deutsch zu verbessern.

Das meta-linguistische Feedback wird als geeignete Rückmeldung für diese Übungen gewählt, weil es mit der Interaktion Mensch-Computer gänzlich konform ist. Der Parser basiert auf einer festen Reihenfolge von Regeln, in denen die Fehler erkannt werden, indem die Wohlgeformtheitsbedingungen innerhalb der Regel kontrolliert werden.
Das System erzeugt danach ein einfach geschriebenes Feedback, das aber tief gehende Kenntnisse der Sprachwissenschaft – bzw. der linguistischen Termini – voraussetzt. Diese Rückmeldungen weisen rein auf die Fehler hin - nur spezifische Fehler werden bestimmt, wie z.B. die Kongruenz zwischen Subjekt und Verb, die Kasusbildung, und die Inversion der Ergänzungen (direktes und indirektes Objekt).
Das System erstellt keine Fehlerkorrektur, da aus didaktischen Gründen eine schlichte Fehlermeldung den Lernenden ausreichend sein sollte, um den Satz selbstständig korrekt zu überarbeiten.
In den vorangehenden Abschnitten wurden schon einige Feedbacks gezeigt. Nachfolgend werden einige Beispiele von dekontextualisierten Rückmeldungen des Systems gelistet:

```
L'accordo Soggetto-Predicato è sbagliato.
Il soggetto è...
Il verbo è...
La posizione del participio passato è sbagliata.
Il predicato è...
Il costituente è...
```

Il caso del complemento indiretto è sbagliato.
L'oggetto indiretto è...
Caso richiesto: dativo

La parola contiene un errore di ortografia o non è presente nel lessico.

Manca un soggetto al nominativo!
Il verbo è...

L'accordo soggetto predicato è giusto.

La preposizione del complemento obliquo è giusta.

5. Schlussfolgerungen und Ausblick

Diese Arbeit hatte das Ziel, den GI-Tutor (GermanItalian Tutor), ein an der Universität Ca' Foscari entwickeltes System zur automatischen Fehleranalyse, vorzustellen. Darüber hinaus sollte geprüft werden, inwieweit das System für die Implementation von Algorithmen zur Fehleranalyse, -erkennung und -meldung geeignet ist.

Eine wichtige Grundbedingung für die Implementierung des Systems war der Algorithmus, der fehlerhafte Sätze bis zum Ende komplett analysieren bzw. parsen konnte.
Im Hinblick auf eine Verwendung als Sprachlehrsystem soll der Parser nicht abbrechen, wenn Fehler im Satz vorhanden sind.
Zwei Problematiken mussten gelöst worden, damit alle verschiedenen Eingabeketten im System bewertet worden konnten: zuerst, die Analyse des sprachlichen Inputs; zweitens, die Fehleranalyse.
Einerseits hatte sich die Analyse des sprachlichen Inputs durch die Verwendung eines extra gebildeten Trainingskorpus ergeben, das eine detaillierte Untersuchung der unterschiedlichen Fehlervarianten erlaubt hat. Anderseits wurde die Fehleranalyse durch einen zweiten Algorithmus realisiert, womit die morpho-syntaktischen und funktionalen Informationen der Wörter und Konstituenten gewonnen werden konnten. Dies erlaubte die Erkennung der Fehler und die darauf folgende Generierung einer angemessenen Rückmeldung.

Für diese Untersuchung wurde das schon vorhandene Lexikon durch ein weiteres Lexikon der Universität Hamburg ergänzt, um adäquater und effizienter zu sein. So wurden also nicht nur Verben, Nomen, Eigennamen, Adjektive, Adverbien usw., die sich in den Sätzen des Trainingskorpus befinden, aufgenommen, sondern auch Tausende von weiteren Einträgen, die alle mit morphologischen, syntaktischen und semantischen Informationen sorgfältig ausgestattet wurden. Dieses ausgedehnte Lexikon erlaubt eine zuverlässige Analyse aller möglichen Eingabeketten der Lernenden.
Nebenher wurde der Parser überarbeitet und implementiert, damit alle Beispielsätze - sowohl die fehlerhaften als auch die richtigen Sätze – analysiert und gute Ergebnisse

geliefert werden konnten.
Alle Sätze wurden auch mit dem Parser ParZu, der am Institut für Computerlinguistik in Zürich entwickelt wurde, geparst. Dadurch konnte die Analysefähigkeit von GI-Tutor unter Verwendung der 1252 Sätze geprüft, und diese Resultate konnten mit denen des ParZu verglichen werden.

Wie Kapitel 4 zeigte, wurden viele unterschiedliche Strukturen im vorliegenden Grammatiksystem behandelt.
Dennoch werden noch nicht alle Konstituenten richtig analysiert, da einige Strukturen, die häufige Fehler verursachen, ausgelassen wurden. In naher Zukunft werden diese unbehandelten Details in die Grammatik integriert. Hierzu zählen beispielsweise die komplexen passiven Verbalstrukturen oder die extraponierten Sätze.
Des Weiteren müssen die Feedbacks für jeden Fehlertyp der Datenbank implementiert werden, sodass die Lernenden eine ausführliche und komplette Meldung bekommen können.

Während des nächsten Studienjahrs wird GI-Tutor den Lernenden auf der Webseite der Universität Ca' Foscari zur Verfügung gestellt. Das System ist zurzeit als freistehende Applikation implementiert und befindet sich auf den Computern des Ca' Foscari Computerlinguistik Labors.
Da das System noch nicht in Verwendung ist, gibt es noch keine Statistik über dessen Verwendung und dessen effektive Hilfe für die Lernenden.
Wenn GI-Tutor in Zukunft aber verwendet wird, können die Wirkungen untersucht werden und das System kann an die Bedürfnisse der Lernenden angepasst werden.

Für die zukünftige Verwendung bzw. Weiterentwicklung des Programms – jedoch außerhalb des Rahmens dieser Arbeit – sollte auch berücksichtigt werden, dass GI-Tutor nicht für Lernende mit starken Augenschwächen konzipiert wurde. Allerdings könnten ein Kontrastinterface und eine synthetische Klangerzeugung in der Benutzeroberfläche angeboten werden, um die Barrierefreiheit des Systems zu erhöhen.

Anhang 1: Das grundlegende Lexikon

In diesem Anhang werden die ersten 8000 Lexikoneinträge des Programms GI-Tutor beschrieben, die als Listen einfach und verständlich organisiert sind.
Wie schon in Kapitel 3 erwähnt, wurde dieses Lexikon durch neue Dateien aus JWCDG Lexikon der Universität Hamburg ausgedehnt. Um die benutzten Dependenzetiketten zu begreifen, sollen Foth (2006) und Foth et al. (2014) konsultiert werden.

Die Wortkategorien des Wortschatzes

Jede Wortform des Lexikons enthält ihr Lemma und entsprechende Eigenschaften, in der Form von Attribut-Werte-Paaren. Wie schon erwähnt, wurde das Lexikon in Rahmen dieser Arbeit implementiert und besteht derzeit aus etwas mehr als 8000 Einträgen: ungefähr 1474 Verbeinträge, 3294 Nomeneinträge, 2293 Adjektiveinträge, etc.
Neuen Wortklassen wurden eingefügt. Nachfolgend werden die Wortkategorien in alphabetischer Reihenfolge erläutert; die Nummer neben jedem Etikett bezeichnet die Anzahl der Argumente der gegebenen Wortform. Diese Tags (Etikette, Labels) stellen Wortklassen, deren Art und Bedeutung die durch morphologische und semantische Eigenschaften bestimmt werden, dar. Je nach der Funktion in einem Satz, kann ein Wort zu mehreren Wortarten gehören.

1. **Adjektive**
   - a_ger/1 beinhaltet alle Adjektive, die sich in den Sätzen des Korpus finden;

```
a_ger(akademisch).
a_ger(leidenschaftslos).
a_ger(produktiv).
```

   - a_ger_ecc/1 listet die Adjektiveinträge, die einen Orthographiefehler aufweisen;

```
a_ger_ecc(correct).
a_ger_ecc(strong).
a_ger_ecc(fruh).
```

- cardin/3 enthält Kardinalzahlen (Zahladjektive), sowohl in Worten als auch in Nummer;

```
cardin(sieben, 7, plur).
cardin(dreizehn, 13, plur).
cardin(neunzehnhundertundfuenfzig, 1950, plur).
```

- ord/4 bezeichnet die Ordinalzahlen (Zahladjektive);

```
ord(siebte, siebt, mas, plur).
ord(dreizehnte, dreizehnt, mas, plur).
ord(neunundneunzigste, neunundneunzigst, mas, plur).
```

- pred_a/3 beinhaltet Adjektive und in Klammern deren semantische Eigenschaften;

```
pred_a(kulturmaessig, [quantif], [sn/sogg/[spazio, tempo,
   misura], sp/obl/misura/a/[spazio, tempo, misura]]).

   pred_a(wahnsinnig,     [soggettivo],     [sn/sogg/[umano,
   animato,oggetto]]).
pred_a(zuverlaessig,     [valutativo],     [sn/sogg/[umano,
   animato]]).
```

2. Artikeln

- art_det/4 bezeichnet die bestimmten Artikel und in Klammern ihre grammatikalischen Eigenschaften;

```
art_det(der, mas, sing, nom).
art_det(die, fem, sing, nom).
art_det(das, neut, sing, nom).
```

- art_ndet/4 stellt die unbestimmten Artikel und ihre grammatikalischen Eigenschaften vor;

```
art_ndet(eines, mas, sing, gen).
art_ndet(einer, fem, sing, dat).
art_ndet(einem, neut, sing, dat).
```

3. Adverbien

- avv/2 beinhaltet die Adverbien und in Klammern ihre semantischen Eigenschaften;

```
avv(dennoch, avv(pred_avv(dennoch), tipo(avvers))).
avv(teilweise, avv(pred_avv(teilweise), tipo(modal))).
avv(vormittags,  avv(pred_avv(vormittags),  tipo(deittico),
   duraz(non_punt), config([td<tr]))).
```

- avv_ecc/2 enthält die Adverbien, die einen Orthographiefehler haben;

```
avv_ecc(here,        avv(pred_avv(here),        tipo(locat),
   spec(def(+)))).
avv_ecc(seher, sehr).
```

- inten/1 sind die Modaladverbien, die einer Äußerung eine besondere Intonation zuweisen;

```
inten(echt).
inten(sehr).
inten(unbeschreiblich).
```

4. Interjektion

- interjc/1 bezeichnet alle Wörter, die in der Alltagssprache als Interjektionen benutzt werden;

```
interjc(aua).
interjc(tja).
interjc(tschuess).
```

5. Interpunktion

- sub_w_dir/1 stellt mögliche Satzzeichen vor;

```
sub_w_dir('?').
sub_w_dir('"').
sub_w_dir('!').
```

6. Konjunktionen

- compl/1 bezeichnet die Konjunktion, welche Dass-Sätze einleitet;

```
compl(dass).
```

- comple/1 beinhaltet die Konjunktionen, welche die Infinitivkonstruktionen kennzeichnen;

```
comple(zu).
comple(um).
comple(ohne).
```

- conju/1 enthält die Konjunktionen, die gleichrangige Sätze in Verbindung setzen;

```
conju(aber).
conju(denn).
```

```
conju(und).
```

- sub_w_sn/2 bezeichnet Einheiten, die Vergleichsätze einführen;

```
sub_w_sn(wie, comparativa).
sub_w_sn(sowie, esplicativa).
```

- sub_w/1 enthält die Konjunktionen, die verschiedene Nebensätze einführen, beziehungsweise kausal, konditional, temporal.

```
sub_w(als).
sub_w(ob).
sub_w(weil).
```

7. Partikeln

- foca/1 bezeichnet Wörter, die als Rangattribute betrachtet werden;

```
foca(auch).
foca(wirklich).
```

- nega/1 enthält die Negationspartikel;

```
nega(nicht).
```

- tren/1 beinhaltet die Partikel der trennbaren Verben;

```
tren(dar).
tren(hinaus).
tren(zurueck).
```

8. Präpositionen

- prepart/3 enthält die mit einem bestimmten Artikel zusammengesetzten Präpositionen;

```
prepart(an, am, dem).
prepart(durch, durchs, das).
prepart(um, ums, das).
```

- prepos/2 beinhaltet Präpositionen und den Kasus, den sie tragen;

```
prepos(an, [dat, acc]).
prepos(seit, dat).
prepos(wegen, gen).
```

- pronavv/3 enthält die mit „da“ zusammengesetzten Präpositionen;

```
pronavv(da, darin, in).
pronavv(da, darunter, unter).
pronavv(da, dazwischen, zwischen).
```

9. Pronomina

- cl/7 beinhaltet Klitika (engl. *clitics*), bzw. Personalpronomina und ihre grammatikalischen Eigenschaften;

```
cl(clitico, _, dich, 2, _, sing, acc).
cl(clitico, _, ihr, 3, fem, sing, dat).
cl(clitico, _, uns, 1, _, sing, [acc, dat]).
```

- pr/7 beinhaltet verschiedenartige Pronomina - beispielsweise impersonale Pronomen, Personalpronomina, Possessivpronomen, deiktische Ausdrücke etc.

```
pr(personale, _, mich, 1, _, sing, acc).
pr(deittico, '_', jene, 3, fem, sing, acc).
pr(pr_quantif, '_', nichts, 3, _, sing, acc).
```

- pred_ap/3 bezeichnet die Possessivpronomina mit Subjektfunktion;

```
pred_ap(mein, _, [sn/sogg/_]).
pred_ap(ihre, _, [sn/sogg/_]).
pred_ap(eur, _, [sn/sogg/_]).
```

- pred_p/5 bezeichnet die Personalpronomina, welche Subjektfunktion tragen;

```
pred_p(pro, wir, nil, [animato, umano], []).
pred_p(pro, es, nil, [animato, luogo], []).
pred_p(pro, sie, nil, [animato, umano], []).
```

- prointerrogativo/4 enthält die Interrogativpronomina und in eckigen Klammern ihre semantischen Eigenschaften;

```
prointerrogativo(wer,[nom],[umano,animato],sn).
prointerrogativo(wohin, [obl], [luogo],sp).
prointerrogativo(woran, [acc,obl], [umano,animato],sp).
```

- quant/5 stellt die indefiniten Pronomen mit ihren grammatikalischen Eigenschaften vor;

```
quant(etwas, etwas, +, neut, sing).
quant(jemandem, jemandem, +, mas, sing).
quant(niemand, niemand, '_', mas, sing).
```

- quant_int/5 stellt nur einen einzigen Typ der Interrogativpronomina vor; die anderen sind in prointerrogativo/4;

```
quant_int(welche, welche, '_', fem, sing).
quant_int(welches, welches, '_', neut, sing).
quant_int(welchen, welchen, '_', _, plur).
```

- relp_d/4 enthält die Relativpronomen und ihre grammatikalischen Eigenschaften;

```
relp_d(der, fem, sing, gen).
relp_d(dem, neut, sing, dat).
relp_d(denen, _, plur, dat).
```

10. Substantive

- alterato/5 bezeichnet die Verkleinerungformen der Nomen;

```
alterato(hauschen, haus, klein, fem, sing).
alterato(liederlein, lieder, klein, fem, sing).
alterato(stueckchen, stuech, klein, mas, plur).
```

- nd/5 beinhaltet die Wortformen der Substantive in dieser Reihenfolge: Nominativ oder Akkusativ Singular, Nominativ oder Akkusativ Plural, Dativ Plural, Genitiv Plural, Genus;

```
nd(bindestrich, bindestriche, bindestrichen, bindestrichs,
   mas).
nd(geschwindigkeit, geschwindigkeiten, geschwindigkeiten, 0,
   fem).
nd(vorlesung, vorlesungen, vorlesungen, 0, fem).
```

- necc/1 listet die Substantive, die einen Orthographiefehler zeigen, auf;

```
necc(halsmerzen).
   necc(schulhe).
   necc(yahre).
```

- pred_n/4 beinhaltet die Substantive und in eckigen Klammern ihre semantischen Eigenschaften;

```
pred_n(artikel, nil, [oggetto, informa], []).
pred_n(russischkurs, nil, [evento, coll, soc, instit], []).
pred_n(zeitspanne, nil, [tempo, misura, nquant], []).
```

- npro/3 enthält Eigennamen und geographische Namen zwischen einfachen Anführungszeichen, zusammen mit ihrem Genus.

```
npro('Sabi', '??', fem).
npro('Venedig', '??', neut).
npro('Martin', '?? ', mas).
```

- pred_np/4 bezeichnet die Eigennamen, geographischen Namen und Zeitangaben; die Namen stehen zwischen einfachen Anführungszeichen und ihren semantischen Eigenschaften in eckigen

Klammern.

```
pred_np('Sabi', nil, [umano], []).
pred_np('Venedig', nil, [luogo], []).
pred_np(1945, nil, [tempo, durata], []).
```

11. Verben

- aux/6 enthält alle konjugierten Hilfs- (haben, sein, werden) und Modalverben (dürfen, können, mögen, müssen, sollen, wollen).

```
aux(bin, sein, ind, pres, 1, sing).
aux(werdest, werden, cond, pres, 2, sing).
aux(duerfen, duerfen, ind, pres, 1, plur).
```

- inf_zu/1 listet die trennbaren Verbeinträge auf, die in ihrer Infinitivform die Partikel „zu“ enthalten;

```
inf_zu(anzukommen).
inf_zu(fernzusehen).
inf_zu(zurueckzufahren).
```

- pp_ger/1 enthält die Partizip-Perfekt Formen der Verben;

```
pp_ger(gelesen, lesen).
pp_ger(geschuetzt, schuetzen).
pp_ger(organisiert, organisieren).
```

- pp_ger_ecc/1 enthält alle Partizip-Perfekt Formen, die einen Konjugations- oder Orthographiefehler zeigen

```
pp_ger_ecc(gefinden, finden).
```

- pred_vc/4 bezeichnet die Verben, die komplexe semantische Eigenschaften aufzeigen; hierzu gehören auch die kopulativen Verben;

```
pred_vc(gehen, inac, camb_attiv, [[sn/sogg/agente/[animato,
   umano],              vcomp/prop/[]/[sogg=sogg/agente]],
   [sn/sogg/agente/[animato,                      umano],
   sp/obl/loc_dir/zu/[umano,                     luogo]],
   [sn/sogg/agente/[animato, umano], sp/obl/goal/[auf, in,
   nach]/[luogo]], [sn/sogg/agente/[animato, umano]]]).

pred_vc(fahren, inac, camb_attiv, [[sn/sogg/agente/[animato,
   umano],              vcomp/prop/[]/[sogg=sogg/agente]],
   [sn/sogg/agente/[animato,                      umano],
   sp/obl/loc_dir/zu/[umano,                     luogo]],
   [sn/sogg/agente/[animato, umano], sp/obl/goal/[auf, in,
   nach]/[luogo]], [sn/sogg/agente/[animato, umano]]]).

pred_vc(haben,  cop,  stato,  [[sn/sogg/tema_bound/[umano,
   animato], ncomp/prop/[_| _]], [sn/sogg/esperiente/[umano,
```

```
animato],            sn/ogg/tema_bound/[_|            _],
pcomp/locativo/[luogo]]]).
```

- pred_vs/14 beinhaltet alle Verben (transitiv, intransitiv, inakkusativ, reflexiv, psychologisch, etc.) - mit Ausnahme der kopulativen Verben;

```
pred_vs(legen, trans, risultato, sn-sogg, esperiente,
[umano], sn-sogg, [nn], esperiente, [umano, animato,
oggetto], pcomp, [um, in, ueber, auf], locativo, [luogo]).

pred_vs(wohnen, intr, stato, sn-sogg, tema_unaff, [umano,
animato], pcomp, [auf, bei, in], locativo, [umano, luogo],
nn, [nn], nn, nn).

pred_vs(vorstellen, rifl, attivita, sn-sogg, agente,
[umano], sn-ogg, [nn], nn, [nn], nn, [nn], nn, nn).
```

- v/6 beinhaltet alle konjugierten Verbformen, ausgenommen die Hilfs- und Modalverben;

```
v(fuehle, fuehlen, ind, pres, 1, sing).
v(lacht, lachen, imperat, pres, 2, plur).
v(schuetzte, schuetzen, ind, past, 3, sing).
```

- vecc/1 enthält alle Verbeinträge, die einen Konjugations- oder Orthographiefehler zeigen;

```
vecc(lieght, liegen).
vecc(kommte, kommen).
vecc(sint, sein).
```

Das Tagset

Ein Tagset stellt ein Inventar an Wortarten und Klassifikatoren dar, die jedes Wort während der Tokenisierung bezeichnet. Je größer ein Tagset ist, desto mehr steigt dir Mehrdeutigkeit bei der Zuweisung der Label und der Tagger kann mithin zahlreiche Fehler produzieren.

Das folgende Tagset ist relativ klein (ungefähr 85 Tags) und es wurde anfänglich für die Analyse von Texten im Englischen und Italienischen gebaut. Daher ist der Großteil der Etikette Englisch oder Italienisch benannt. Spezifische Tags für das Deutsche wurden jedoch schon von Marino (2000/01) weiter entwickelt.

Tag	Bedeutung
ag	Adjektiv
art	bestimmter oder unbestimmter Artikel
auag	Gerundiv des Hilfsverbs „haben“
auair	Hilfsverb „haben“ (irreale Modalität)
aueg	Gerundiv des Hilfsverbs „sein“
aueir	Hilfsverb „sein“ (irreale Modalität)
ausa	konjugiertes Hilfsverb „haben“
ausai	infinites Hilfsverb „haben“
ausap	Partizip Perfekt des Hilfsverbs „haben“
ause	konjugiertes Hilfsverb „sein“
ausei	infinites Hilfsverb „sein“
ausep	Partizip Perfekt des Verbs „sein“
ausp	konjugiertes Hilfsverb „werden“
auspir	Hilfsverb „werden“ (irreale Modalität)
avv	Adverb
avvd	zusammengesetzte Präposition (da + Präposition)
clit	pronominales Klitikon
clitac	pronominales Klitikon, Kasus Akkusativ
clitdat	pronominales Klitikon, Kasus Dativ
cong	Konjunktion
cosu	Subordinierende Konjunktion
deit	deiktisches Pronomen
dim	Demonstrativadjektive
fw	Fremdwort
in	Intensifikator, Steigerungspartikel
ind	Unbestimmtes Adjektiv
infzu	infinitives Verb mit „zu“
int	Interrogativ
intj	Interjektion
n	Nomen
na	Diminutivum, Verkleinerungsform (Nomen)
neg	Negation, Verneinung

np	Eigennamen
num	Numerale, Zahlwort
p	Präposition
part	Zusammengesetzte Präposition
pers	Personalpronomen
pk	Komplementierer
poss	Possessivpronomen
pron	Pronomen
pt	Verbalpartikel
pt1	Infinitivpartikel „um“, „statt“, „ohne“
pt2	Infinitivpartikel „zu“
punt	Satzzeichen nicht am Satzende
puntint	nicht deklaratives Satzzeichen
punto	Satzzeichen am Ende des Satzes
q	Quantifikator
rel	Relativsatz
tren	trennbare Partikel
vc	Kopulativverb
vci	infinitives oder konjugiertes kopulatives Verb „werden“
vcir	Kopulativverb (Irrealis)
vgin	Gerundiv eines intransitiven Verbs
vgprog	Progressiv eines intransitiven Verbs
vgs	Gerundiv
vgsf	Verbkonstruktion mit Gerundiv
vgt	Gerundiv eines transitiven Verbs
viin	Infinitiv eines intransitiven Verbs
vin	konjugiertes intransitives Verb
vinimp	Imperativ eines intransitiven Verbs
virin	intransitives Verb (irreale Modalität)
virt	transitives Verb (irreale Modalität)
vit	Infinitiv eines transitiven Verbs
vmod	Modalverb
vmodir	Modalverb (Irrealis)
vppc	Partizip Perfekt eines kopulativen Verbs
vppin	Partizip Perfekt eines intransitiven Verbs

vppt	Partizip Perfekt eines transitiven Verbs
vprin	Partizip Präsens eines intransitiven Verbs
vprog	progressives Verb
vprogir	progressives Verb (irreale Modalität)
vprt	Partizip Präsens eines transitiven Verbs
vsupp	Modalverb
vt	konjugiertes transitives Verb
vtimp	Imperativ eines transitiven Verbs

Die Label der Netzwerke

Um das Programm sorgfältig zu verstehen, werden die Label der in GI-Tutor implementierten Netzwerke aufgeführt. In den Eingangsknoten beinhalten die Netzwerke nur pre-terminale Symbole, um die Mehrdeutigkeit zu verkleinern.

Tag	**Bedeutung**	**Beispiel**
cp	Ergänzungssatz	tp_access(cp, [avv, avvl, ccom, congf, cong, conjl, cosu, fine, int, in, intj, neg, p, part, puntint, punto, tren, vinimp, vtimp]).
f	Satz	tp_access(f, [ag, agn, art, ausa, ausair, ause, auseir, clit, clitac, clitdat, date, deit, dim, in, ind, n, noun, na, neg, nf, nh, np, num, p, par, pers, pk, poss, pron, puntint, punto, q, qc, qd, relin, vc, vcir, vin, virin, virt, vppt, vprog, vprogir, vsupir, vsupp, vt]).
f2	relativer Präpositionalsatz	tp_access(f2, [p, rel, relin, relob]).
fac	Konjunktionalsatz	tp_access(fac, [pk]).
fc	abhängiger Satz	tp_access(fc, [congf, cong, avv, np])
fc2	abhängiger Satz (Asyndeton)	tp_access(fc2, [art, pers, pron]).
fimp	Imperativsatz	tp_access(fimp, [avv, p, poss, n, vinimp, vtimp]).
fint	Interrogativsatz	tp_access(fint, [avv, int, intj, n, p, punt, puntint,

		punto]).
fp	parenthetischer Satz	tp_access(fp, [par, punt]).
fs	Nebensatz	tp_access(fs, [conjl, cosu, in]).
fyesno	Interrogativsatz	tp_access(fyesno, [ag, agn, art, date, deit, dim, in, n, noun, neg, nf, nh, np, nt, num, pers, poss, p, pron, puntint, punto, qc, vppt, ausa, ause, ausd, clit, clitac, clitdat, neg, vc, vin, vprog, vsupp, vt]).
ibar	konjugierte Verbalphrase	tp_access(ibar, [ausa, ause, ausp, ausep, ausap, neg, pron, puntint, punto, vc, vin, vprog, vsupp, vt]).
ibar2	konjugierte Verbalphrase	tp_access(ibar2, [vc, ause, ausa, vit, vci, viin, vt, vppin, vppt, vin, vsupp, ausap, ausep]).
ibar3	konjugierte Verbalphrase	tp_access(ibar3, [ag, agn, art, date, deit, dim, in, n, noun, neg, nf, nh, np, na, nt, num, pers, poss, pron, p, part, qc, vit, vci, viin, vt, vin]).
ibar4	konjugierte Verbalphrase	tp_access(ibar4, [ag, ause, ausa, ausei, ausai, agn, art, date, deit, dim, in, n, noun, neg, nf, nh, np, na, nt, num, pers, poss, pron, p, part, qc, vit, vci, viin, vt, vin, vppin]).
ibar5	konjugierte Verbalphrase	tp_access(ibar5, [ausa, ause, ausp, ausep, ausap, neg, pron, puntint, punto, vc, vin, vprog, vsupp, vt, in, p, part, php]).
ir_infl	konjugierte Verbalphrase Konjunktiv	tp_access(ir_infl, [ausair, auseir, ausep, ausap, clit, clitac, neg, vcir, virin, virt, vprogir, vsupir]).
ogg	Akkusativobjekt	tp_access(ogg, [ag, agn, art, date, deit, dim, in, n, noun, na, neg, nf, nh, np, nt, num, pers, poss, pron, qc, vppt]).
ogg1	Akkusativobjekt	tp_access(ogg1, [ag, agn, art, clitac, clitdat, date, deit, dim, in, n, noun, na, neg, nf, nh, np, nt, num, pers, poss, pron, qc, vppt]).
ogg2	Akkusativobjekt	tp_access(ogg2, [ag, agn, art, date, deit, dim, in, n, noun, na, neg, nf, nh, np, nt, num, pers, poss, pron, qc, vppt]).

sa saa	Adjektivphrase	tp_access(sa, [ag, agn, avv, ccom, in, ind, neg]).
savv	Adverbialphrase	tp_access(savv, [avv, avvd, avvl, in, intj, nt, p]).
sn	Nominalphrase	tp_access(sn, [ag, agn, art, clitdat, clitac, date, deit, dim, in, n, necc, noun, nf, nh, np, na, nt, num, pers, poss, pron, q, qc, vppt]).
snsempl	Einfache Nominalphrase	tp_access(snsempl, [ag, agn, art, clitac, clitdat, date, deit, dim, in, n, noun, neg, nf, nh, np, na, nt, num, pers, poss, pron, qc, vppt]).
sogg	Subjekt-Phrase	tp_access(sogg, [ag, agn, art, avv, date, deit, dim, in, n, na, noun, neg, nf, nh, np, nt, num, pers, poss, pron, qc, vppt]).
sogg2	Subjekt-Phrase mit einer Adjunkt-PP	tp_access(sogg2, [ag, agn, art, date, deit, dim, in, n, na, noun, neg, nf, nh, np, nt, num, pers, poss, pron, qc, vppt]).
sp	Präpositionalphrase	tp_access(sp, [in, p, part, php]).
sp2	Koordinierte Prepositionalphrase	tp_access(sp2, [art, in, p, part, php, poss]).
spd	Prepositionalphrase (di/da)	tp_access(spd, [in, partd, partda, pd, pda]).
sq	quantifizierende Phrase	tp_access(sq, [in, ind, q, qc, qd]).
sv2	Adjunkt-Satz	tp_access(sv2, [ausai, ausei, ag, agn, clitdat, clitac, date, deit, dim, in, infzu, n, noun, na, neg, nf, nh, np, nt, num, pt1, pt2, p, qc, deit, vppt, viin, vit, vci]).
sv3	partizipialer Adjunkt-Satz	tp_access(sv3, [vppin, vppt, vprin, vprt, avv]).
sv5	gerundiver Adjunkt-Satz	tp_access(sv5, [auag, aueg, vgin, vgs, vgt, vsfa]).
svc	kopulative Verlbalphrase (Kopula)	tp_access(svc, [ag, agn, art, avv, avvd, ccom, clitac, clitdat, date, deit, dim, in, ind, n, noun, na, neg, nf, nh, np, num, p, part, pers, php, poss, pron, qc, tren, vppt, vppin]).
svin	intransitive Verbalphrase	tp_access(svin, [ag, art, ausai, ausei, avv, avvl, clitac, clitdat, ccom, congf, in, ind, int, intj, neg, p,

		`part, php, pk, pt, tren, viin, vit, vci]).`
`svpass`	Passivsatz	`tp_access(svpass, [ag, agn, art, ausai, ausei, avv, avvl, clitac, clitdat, date, deit, dim, dirs, in, ind, int, intj, n, noun, na, neg, nf, nh, np, num, p, par, part, partd, partda, pd, pda, pers, php, pk, poss, pron, pt, punt, q, qc, qd, relin, viin, vit, vci, vppt, vppc]).`
`svt`	transitive Verbalphrase	`tp_access(svt, [ag, agn, art, ausai, ausei, avv, avvl, clitac, clitdat, date, deit, dim, dirs, in, ind, int, intj, n, noun, neg, nf, nh, np, num, p, par, part, partd, partda, pd, pda, pers, php, pk, poss, pron, pt, punt, q, qc, qd, relin, tren, viin, vit, vci, vppt]).`
`yesno`	Ja/Nein-Frage Konjugierte Verbalphrase von `fyesno`	`tp_access(yesno, [ausd, ausa, auspir, vsupp, ause, int, intj, ag, agn, art, date, deit, dim, in, n, noun, neg, nf, nh, np, nt, num, pers, poss, pron, qc, vppt, ausa, ause, ausd, clit, clitac, clitdat, neg, vc, vin, vprog, vsupp, vt]).`
`yesno2`	Konjugierte Verbalphrase	`tp_access(yesno2, [ausd, ausa, auspir, vsupp, ause, int, intj, ag, agn, art, date, deit, dim, in, n, noun, neg, nf, nh, np, nt, num, pers, poss, pron, qc, vppt, ausa, ause, ausd, clit, clitac, clitdat, neg, vc, vin, vprog, vsupp, vt]).`
`yesno3`	Konjugierte Verbalphrase (Imperativ)	`tp_access(yesno3, [vinimp, vtimp]).`

Anhang 2

In diesem Anhang werden die Grafiken der Analyse wiedergegeben.

Prozentsätze der richtigen Analyse aller Sätze		
	GI-TUTOR	**PARZU**
Orthographische Fehler	71,33%	77,33%
Lexikalische Fehler	72,83%	90,76%
Morphologische Fehler	81,19%	80,31%
Syntaktische Fehler	72,32%	79,18%
	74.41%	**81,89%**

Tabelle 2: Prozentsätze der Ergebnisse.

		GI-Tutor		**ParZu**	
		Falsche Analyse	**Richitge Analyse**	**Falsche Analyse**	**Richtige Analyse**
Orthographie					
Fehlerhafte Sätze	75	30	42	27	48
Richtige Sätze	75	10	65	7	68
		28,66%	71,33%	22,66%	77,33%
Lexik					
Fehlerhafte Sätze	92	23	69	16	76
Richtige Sätze	92	27	65	1	91
		27,17%	72,82%	9,23%	90,76%
Morphologie					
Fehlerhafte Sätze	226	47	179	81	145
Richtige Sätze	226	38	188	8	218
		18,80%	81,19%	19,69%	80,31%
Syntax					
Fehlerhafte Sätze	233	69	164	79	154
Richtige Sätze	233	60	173	18	215
		27,68%	72,32%	20,81%	79,18%

Tabelle 3: Resultate.

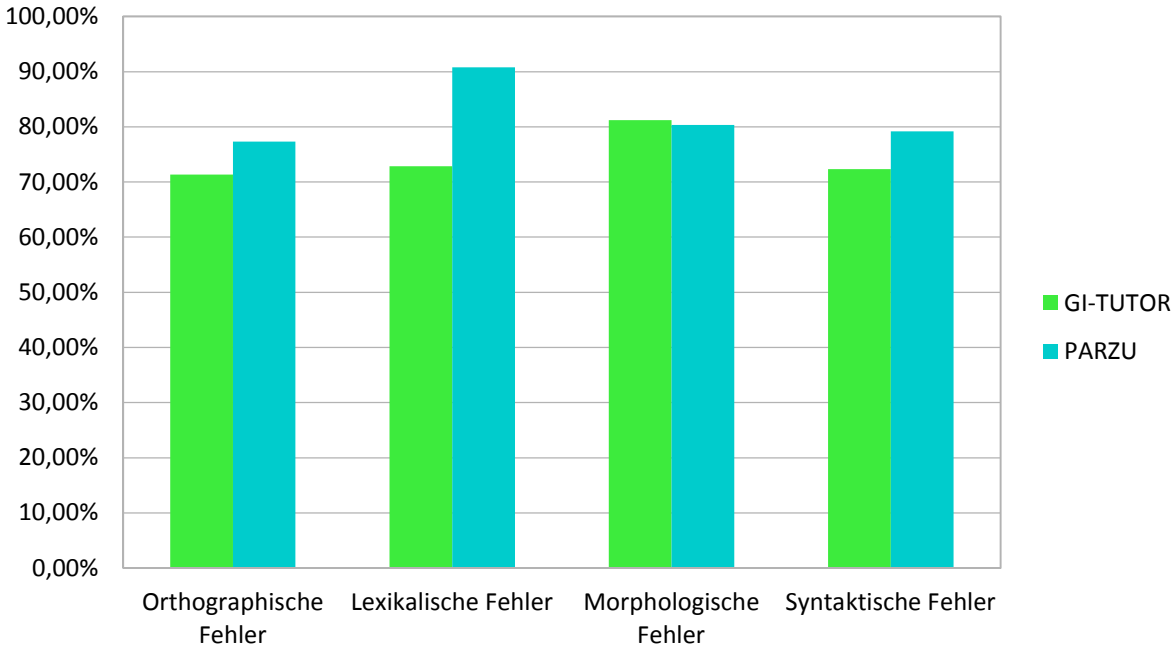

Abb. 14: Grafik 1.

Die folgenden Diagramme zeigen die Ergebnisse, die in der Analyse der fehlerhaften Sätze erzielt wurden.

Orthographische Fehler	GI-Tutor	ParZu
	Korrekte Analyse in %	Korrekte Analyse in %
Othographische Fehler	52,77%	58,33%
Interferenz aus dem Englischen	58,33%	69,44%
Interferenz aus dem Italienischen	66,66%	66,66%
Total	59,25%	64,00%

Tabelle 4: Ergebnisse der korrekten Analyse der Orthographiefehler.

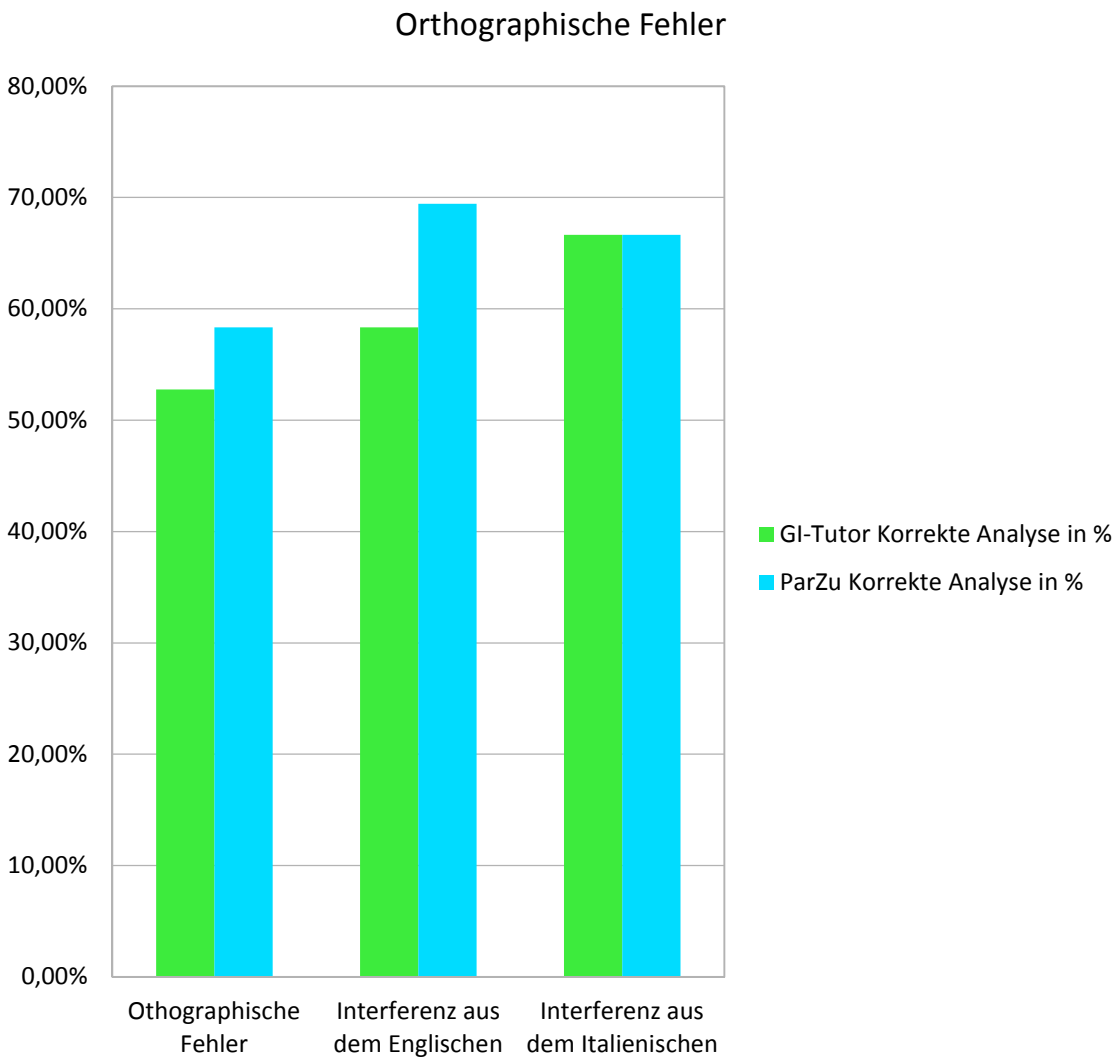

Abb. 15: Grafik 2.

Lexikalische Fehler	GI-Tutor	ParZu
	Korrekte Analyse in %	Korrekte Analyse in %
Selektionsfehler der Präposition	75,86%	94,83%
Selektionsfehler von “Zu”	61,54%	30,77%
Unpassende Hilfsverben	100,00%	100,00%
Unpassende Semantik	75,00%	75,00%
Total	80,23%	88,37%

Tabelle 5: Ergebnisse der korrekten Analyse der lexikalischen Fehler.

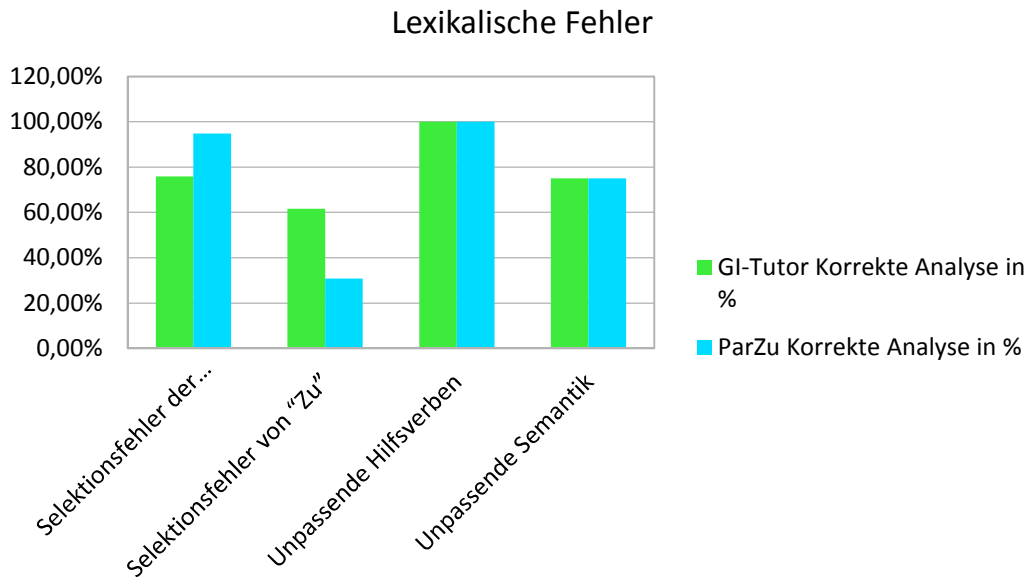

Abb. 16: Grafik 3.

Morphologische Fehler	GI-Tutor	ParZu
	Korrekte Analyse in %	Korrekte Analyse in %
Kongruenzfehler N P Ad Art	81,76%	77,51%
Kongruenzfehler V	86,05%	9,30%
Konjugationsfehler der Vergangenheit	28,57%	64,28%
Total	79,20%	63,71%

Tabelle 6: Ergebnisse der korrekten Analyse der morphologischen Fehler.

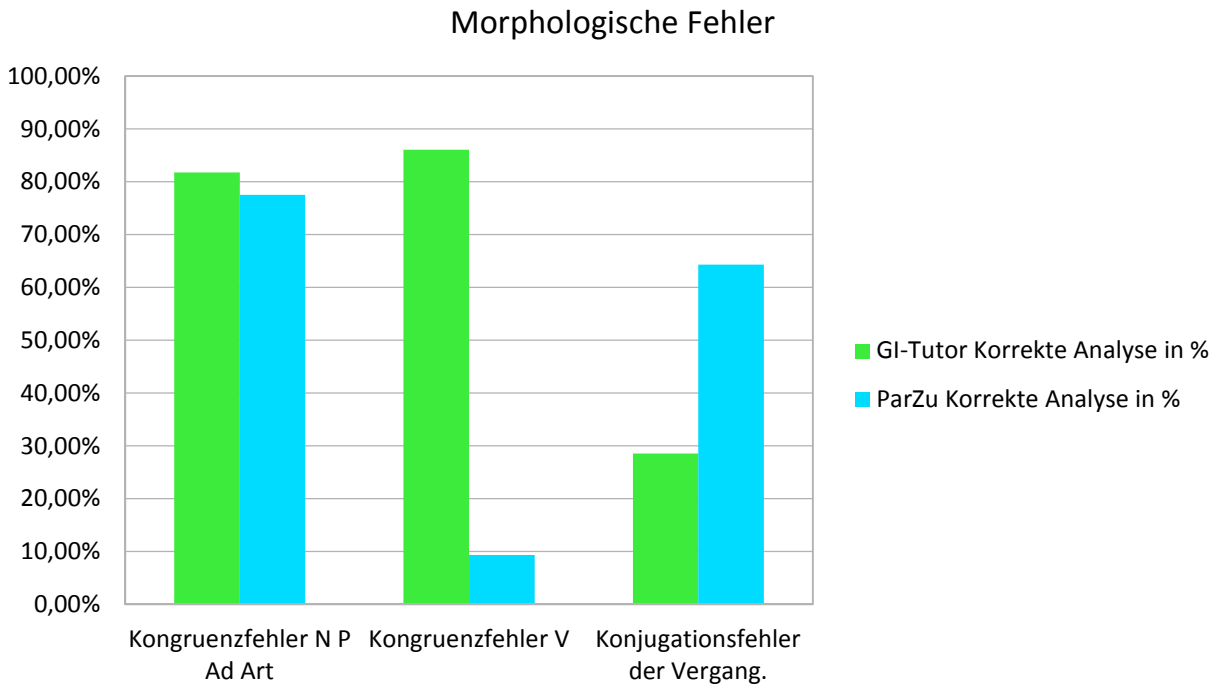

Abb. 17: Grafik 4.

Syntaktische Fehler	GI-Tutor	ParZu
	Korrekte Analyse in %	Korrekte Analyse in %
Verletzung der Satzstruktur	52,17%	60,87%
Verletzung der V2-Regel	73,53%	44,18%
Stellungsfehler im MF	92,31%	84,61%
Stellungsfehler “Nicht”	78,95%	86,84%
Unpassende Verbteile	75,00%	84,37%
Total	70,39%	66,09%

Tabelle 7: Ergebnisse der korrekten Analyse der syntaktischen Fehler.

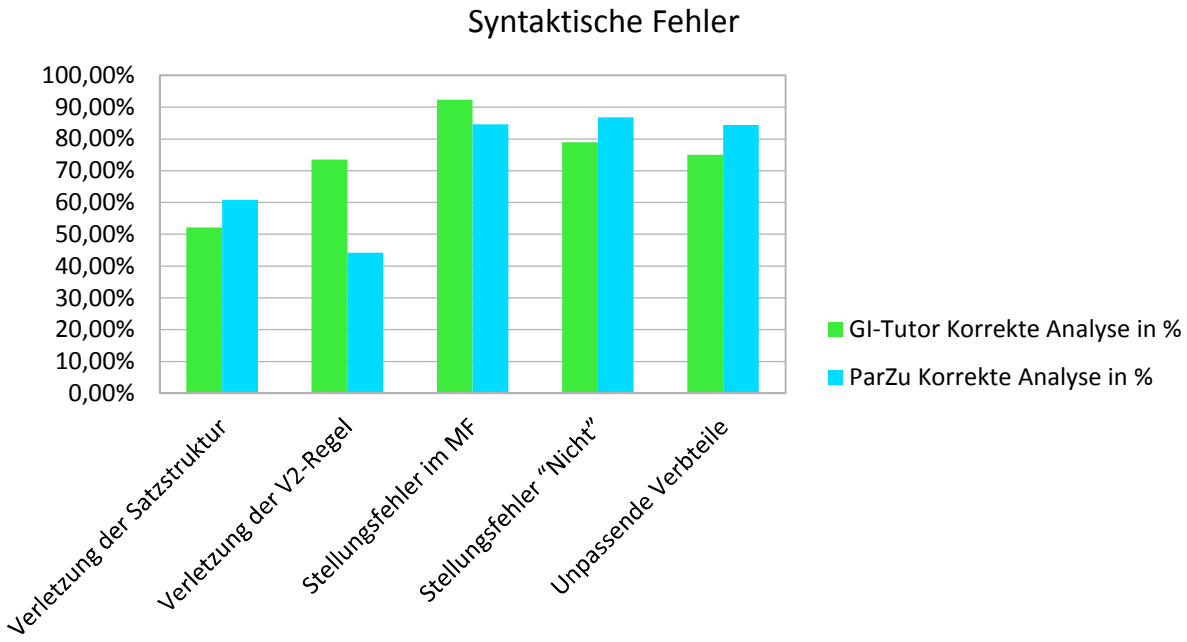

Abb. 18: Grafik 5.

Die folgenden Diagramme zeigen die Ergebnisse, die in der Analyse der wohl geformten Sätze erzielt wurden.

Orthographie	GI-Tutor	ParZu
	Korrekte Analyse in %	Korrekte Analyse in %
Othographische Fehler	97,20%	91,66%
Interferenz aus dem Englischen	80,55%	94,40%
Interferenz aus dem Italienischen	33,33%	33,33%
Total	86,66%	90,66%

Tabelle 8: Ergebnisse der korrekten Analyse der wohl geformten Sätze.

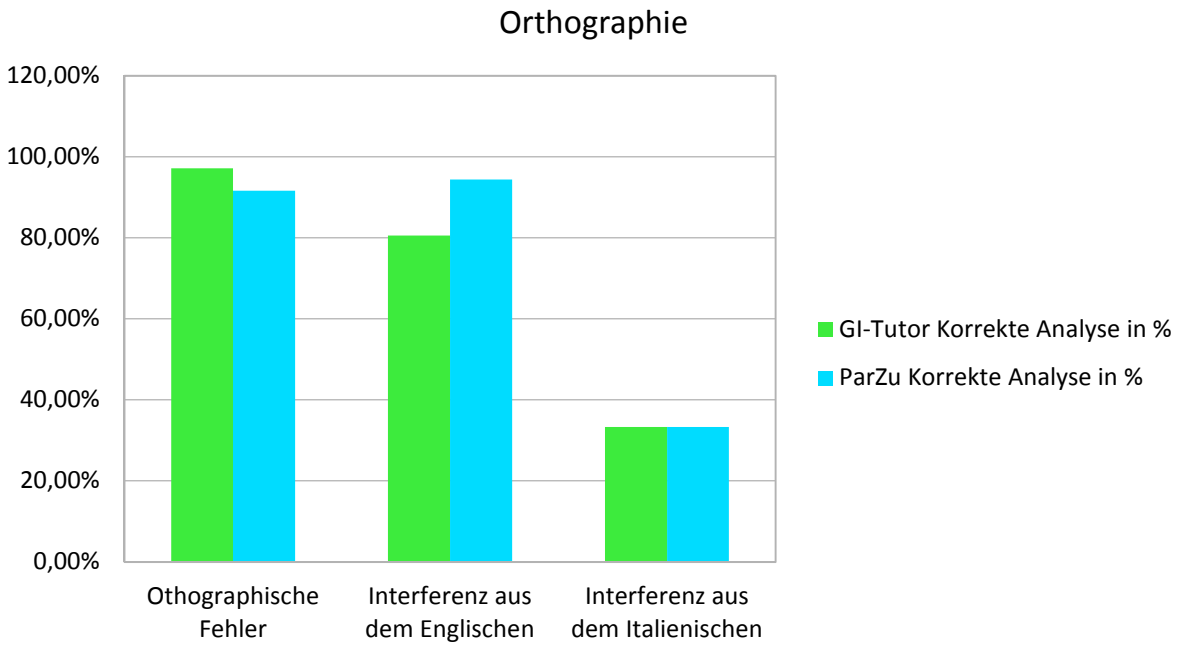

Abb. 19: Grafik 6.

Lexik	GI-Tutor	ParZu
	Korrekte Analyse in %	Korrekte Analyse in %
Selektionsfehler der Präposition	77,59%	100,00%
Selektionsfehler von “Zu”	46,16%	92,31%
Unpassende Hilfsverben	100,00%	100,00%
Unpassende Semantik	56,25%	100,00%
Total	70,65%	98,91%

Tabelle 9: Ergebnisse der korrekten Analyse der wohl geformten Sätze.

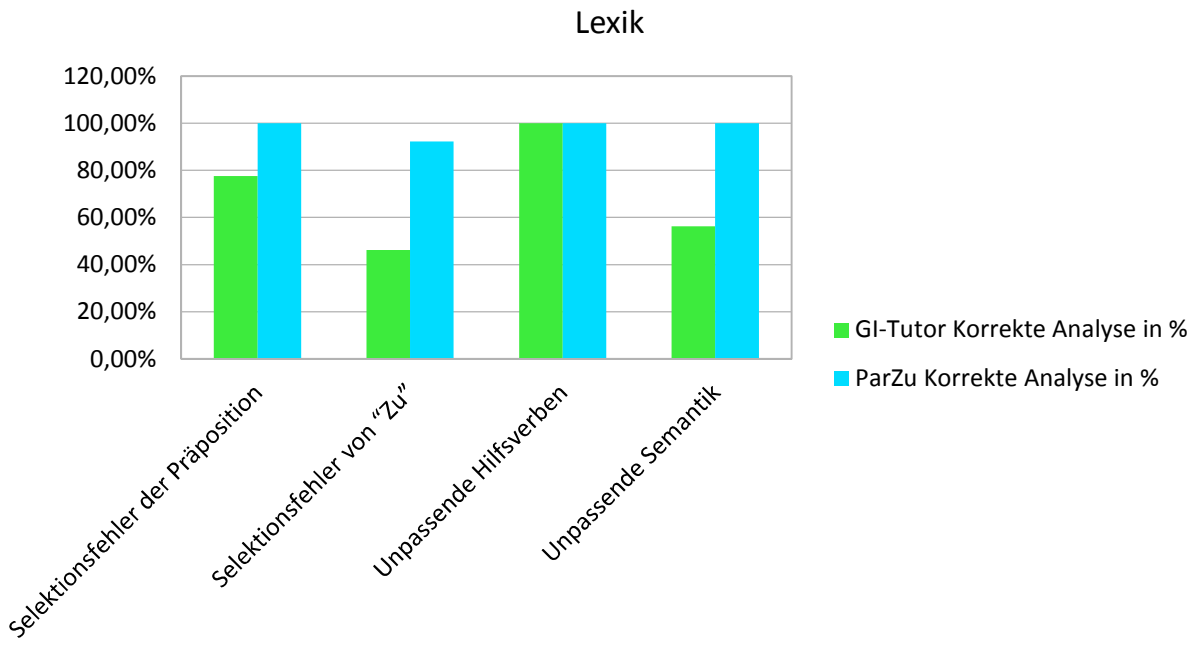

Abb. 20: Grafik 7.

Morphologische Fehler	GI-Tutor	ParZu
	Korrekte Analyse in %	Korrekte Analyse in %
Kongruenzfehler N P Adj Art	82,25%	97,63%
Kongruenzfehler V	81,40%	90,69%
Konjugationsfehler der Vergang.	100,00%	100,00%
Total	83,19%	96,46%

Tabelle 10: Ergebnisse der korrekten Analyse der wohl geformten Sätze.

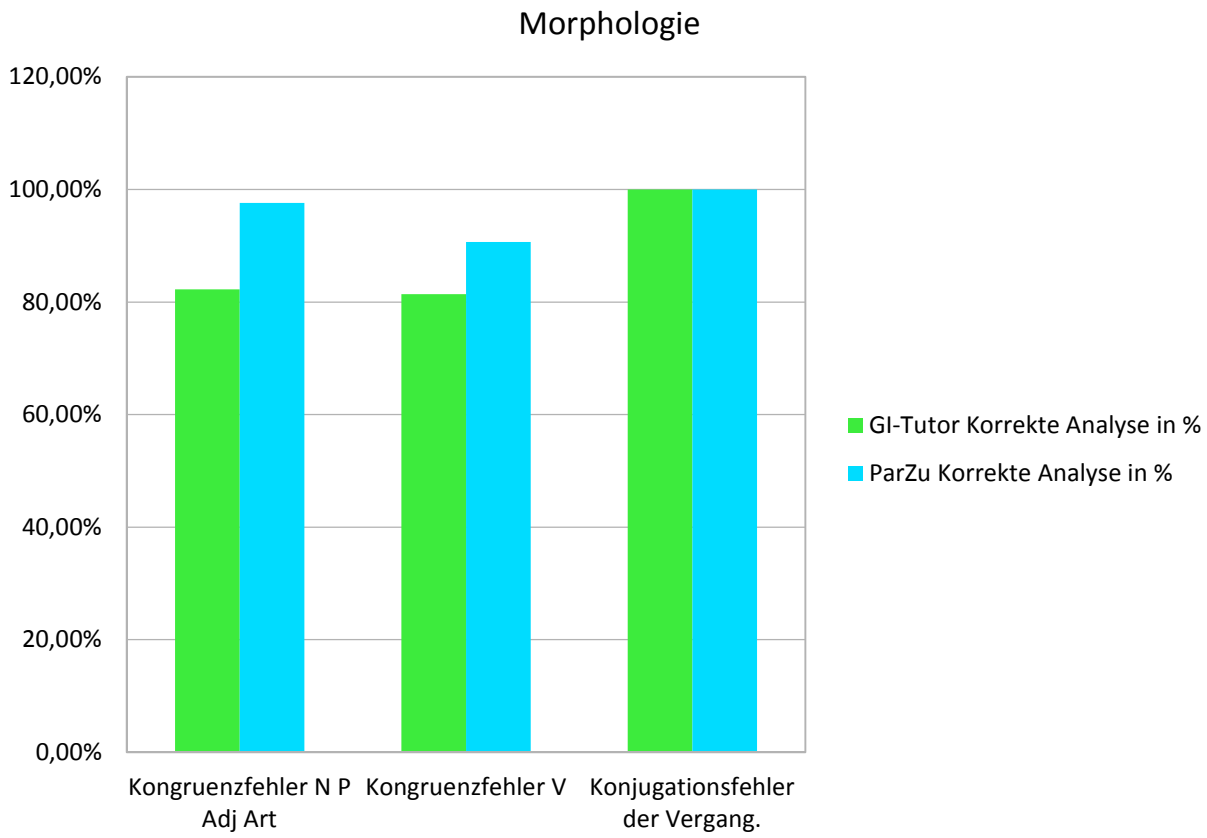

Abb. 21: Grafik 8.

Syntax	GI-Tutor	ParZu
	Korrekte Analyse in %	Korrekte Analyse in %
Verletzung der Satzstruktur	69,56%	89,85%
Verletzung der V2-Regel	72,06%	89,70%
Stellungsfehler im Mittelfeld	88,46%	96,15%
Stellungsfehler “Nicht”	78,95%	94,74%
Unpassende Verbteile	71,87%	96,87%
Total	74,25%	92,27%

Tabelle 11: Ergebnisse der korrekten Analyse der wohl geformten Sätze.

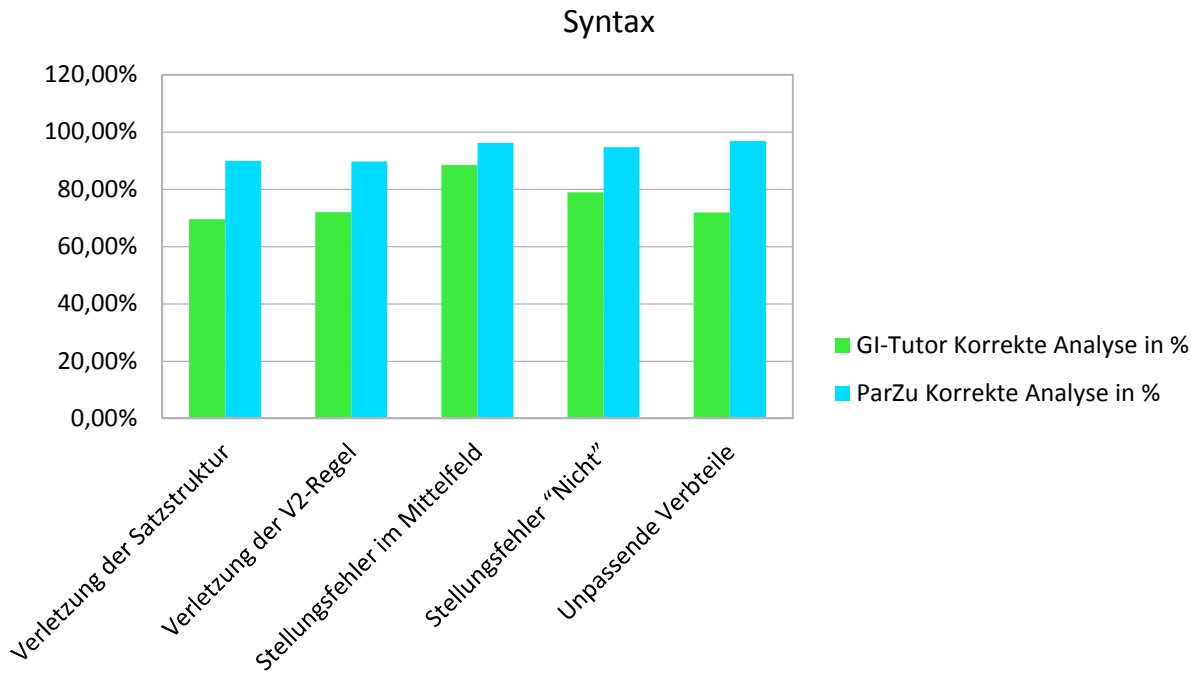

Abb. 22: Grafik 9.

Literaturverzeichnis

Abney, S. P. (1989). A computational model of human parsing. In: *Journal of Psycholinguistic Research,* 18, S. 129–144.

Allen, J. R. (1996, 1997). Ten Desiderata for Computer-Assisted Language Learning Programs. In: *Computers and the Humanities,* 30(6), S. 441-455.

Al-Seghayer, K. (2001). The effect of multimedia annotation modes on vocabulary acquisition: a comparative study. In: *Language Learning & Technology,* 5(1), S. 202–232.

Amaral, L., Meurers, D. (2011). On using intelligent computer-assisted language learning in real-life foreign language teaching and learning. In: *ReCALL*, 23(1), S. 4–24.

Annett, J. (1969). Feedback and Human Behaviour. In: *The effects of knowledge of results, incentives and reinforcement on learning and performance*. Middlesex: Penguin Books Ltd.

Antoniadis, G., Granger, S., Kraif, O., Ponton, C., Zampa, V. (2011). NLP and CALL: Integration is working. In: N. Kübler (Hrsg.) *Corpora, Language, Teaching, and Resources: From Theory to Practice. Selected papers from TaLC7, the 7th Conference of Teaching and Language Corpora*. Bruxelles: Peter Lang.

Ballestreros, M., Nivre, J. (2012). MaltOptimizer: A system for MarltParser optimization. In: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12),* Istanbul.

Bax, S. (2003). CALL—past, present and future. In: *System,* 31(1), S. 13–28.

Beatty, K. (2003). Teaching and Researching Computer Assisted Language Learning. New York: Longman.

Boris, L. (2002). What have you done for me lately? The fickle alignment of NLP and CALL. In: *EuroCALL pre-conference workshop on NLP in CALL, 14. August 2002*, Jyväskylä.

Bresnan, J. (1982). The Mental Representation of Grammatical Relations. Cambridge, Mass: MIT Press.

Bresnan, J., Kaplan, R. (1982). Lexical-functional Grammar, a Formal System for Grammatical Representation. In: J. Bresnan (Hrsg.) *The Mental Representation of Grammatical Relations.* Cambridge, Mass: MIT Press, S. 173-281.

Brown, H. D. (1980). Principles of Language Learning and Teaching. London: Prentice Hall Inc.

Burston, J. (2001). Exploiting the potential of a computer-based grammar checker in

conjunction with self-monitoring strategies with advanced level students of French. In: *CALICO Journal*, 18(3), S. 499–515.

Bußmann, H. (2008). Lexikon der Sprachwissenschaft. Stuttgart: Alfred Kröner Verlag Stuttgart.

Calico, Eurocall & IALLT (1999). Scholarly Activities in Computer-Assisted Language Learning: Development, Pedagogical Innovations, and Research. http://www.calico.org/CALL_document.html <zuletzt abgerufen am 4.06.2015>

Cardinaletti, A., Giusti, G. (1996). Problemi di sintassi tedesca. Padova: Unipress.

Carstersen, K., Ebert, Ch., Ebert, C., Jekat, S., Klabunde, R., Langer, H. (2010). Computerlinguistik und Sprachtechnologie. Eine Einführung. (3. Auflage). Heidelberg: Spektrum Akademischer Verlag.

Carstersen, K. (2012). Sprachtechnologie. Ein Überblick. http://www.kai-uwe-carstensen.de/Publikationen/Sprachtechnologie.pdf <zuletzt abgerufen am 4.06.2015>

Chanier, T., Renié, D., Fouqueré, C. (1993). Actes du colloque SCIAL'93 (sciences cognitives, informatique et apprentissage des langues). Clermont-Ferrand: Université Blaise Pascal.

Chapelle, C. (1998). Multimedia CALL: Lessons to be Learned from Research on Instructed SLA. In: *Language Learning & Technology*, 2(1), S. 22-34.

Chomsky, N. (1965). Aspects of the Theory of syntax. Cambridge: Mass.

Chomsky, N. (1981). Lectures on Government and Binding. Dordrecht: Holland Foris.

Chomsky, N. (1986). Knowledge of Language: Its Nature, Origin, and Use. New York: Praeger.

Chun, D., Plass, J. (1996). Effects of multimedia annotations on vocabulary acquisition. In: *Modern Language Journal,* 80, S.183–198.

Chun, D. (2006). CALL technologies for L2 reading. In: L. Ducate, N. Arnold (Hrsg.) *Calling on CALL: From Theory and Research to New Directions in Foreign Language Teaching.* San Marcos, Texas: CALICO, S. 69–98.

Colmarauer, A. (1978). Metamorphosis Grammars. In: L. Bolc (Hrsg.) *Natural Language Communication with Computers*. Berlin: Springer Verlag, S. 133-189.

Cook, V., Newson, M. (1996). Chomsky's Universal Grammar. An introduction. Oxford: Blackwell Publishers.

Corder, P. (1974). Error analysis. In: J. P. B. Allen, P. Corder (Hrsg.) *The Edinburgh course in applied linguistics: Vol. 3. Techniques in applied linguistics*. London: Oxford

University Press.

Corder, S. P. (1981). Error Analysis and Interlanguage. Oxford: Oxford University Press.

Covington, M. A. (2001). A Fundamental Algorithm for Dependency Parsing. In: J. A. Miller, J. W. Smith (Hrsg.) *Proceedings of the 39th Annual ACM Southeast Conference 2001*. Association for Computing Machinery, S. 95–102.

Crowder, N. (1963). On the difference between linear and intrinsic programming. In: *Phi Delta Kappan*, 44, S. 250-254.

Delmonte, R. (1999). From Shallow Parsing to Functional Structure. In: *Atti del Workshop AI*IA - "Elaborazione del Linguaggio e Riconoscimento del Parlato"*, IRST Trento, S. 8-19.

Delmonte, R. (2000). Parsing Preferences and Linguistic Strategies. In: *LDV-Forum - Zeitschrift für Computerlinguistik und Sprachtechnologie - "Communicating Agents"*, Band 17, 1,2, S. 56-73.

Delmonte, R., Chiran, L., Bacalu, C. (2001). How to integrate linguistic information in files and generate feedback for grammar errors. In: *Workshop on Sharing Tools and Resources for Research and Education, ACL, vol. 1, S. 6-6.*

Delmonte, R. (2002a). GETARUN PARSER - A parser equipped with Quantifier Raising and Anaphoric Binding based on LFG. In: *Proceedings LFG2002 Conference*, Athens, S. 130-153.

Delmonte, R. (2002b). From Deep to Shallow Anaphora Resolution: What Do We Lose, What Do We Gain. In: *Proceedings International Symposium RRNLP.* Alicante, S. 25-34.

Delmonte, R. (2002c). From Deep to Shallow Anaphora Resolution. In: *Proceedings DAARC2002, 4th Discourse Anaphora and Anaphora Resolution Colloquium.* Lisbona, S. 57-62.

Delmonte, R. (2003). Linguistic Knowledge and Reasoning for Error Diagnosis and Feedback Generation. In: *CALICO Journal*, 20(3), S. 513-532.

Delmonte, R. (2005). Deep & Shallow Linguistically Based Parsing. In: A. M. Di Sciullo (Hrsg.) *UG and External Systems.* Amsterdam/Philadelphia: John Benjamins, S. 335-374.

Delmonte, R. (2007). Computational Linguistic Text Processing: Logical Form, Semantic Interpretation, Discourse Relation and Question Answering. New York: NOVA Science Publishers, Inc.

Delmonte, R. (2008). Computational Linguistic Text Processing: Lexicon, Grammar, Parsing and Anaphora Resolution. New York: NOVA Science Publishers, Inc.

Delmonte, R. (2014). A Reevaluation of Dependency Structure Evaluation. In: R. Basili, A. Lenci, B. Magnini (Hrsg.) *Proceedings of the First Italian Conference on Computational Linguistics CLiC-it 2014 & the Fourth International Workshop EVALITA 2014, 9-11 dicembre 2014.* Pisa University Press srl, S. 151-157.

De Marneff, M. C., MacCartney, B., Manning, C. D. (2011). Generating typed dependency parses from phrase structure parses. In: *Proceedings of LREC 2006/5,* S. 449-454.

De Montmollin, M. (1971). L'enseignement programmé. Paris: PUF.

DeRidder, I. (2002). Visible or invisible links: does the highlighting of hyperlinks affect incidental vocabulary learning', text comprehension, and the reading process?. In: *Language Learning & Technology,* 6(1), S. 123–46.

Deville, G., Dumortier, L. (2004). Evaluation d'un outil en ligne d'aide à la lecture de textes en langue étrangère. Actes de la j*ournée d'étude de l'ATALA*, TAL & Apprentissage des langues. Grenoble, S. 93-102.

Dickinson, M. (2006). Writer's aids. In: K. Brown (Hrsg.) *Encyclopedia of language and linguistics* (2. Auflage). Oxford: Elsevier.

Dini, L., Malnati, G. (1993). Weak Constraints and Preference Rules. In: P. Bennett, P. Paggio (Hrsg.) *Preference in Eurotra*. Luxembourg: Commission of the European Communities, S. 75-90.

Dodigovic, M. (2005). Artificial Intelligence in Second Language learning: Raising Error Awareness. Second Language Acquisition Series. Clevedon: Multilingual Matters Ltd.

Dulay, H. C., Burt, M. K. (1974). You Can't Learn without Goofing. In: J. C. Richards (Hrsg.) *Error Analysis: Perspectives on Second Language Acquisition*. London: Longman, S. 95-123.

Dulay, H. C., Burt, M. K., Krashen, S. D. (1982). Language Two. Oxford: Oxford University Press.

Ellis, R. (1994). The study of second language acquisition. Oxford: Oxford University Press.

Eurocall, Calico & IALLT (1999). Eurocall Research Policy Statement. http://www.eurocall-languages.org/ <zuletzt abgerufen am 04.06.2015>

Ferreira, A., Banados, E. (2005). Effective-Corrective Feedback Strategies in Second Language Teaching with Implications for Intelligent Tutorial Systems (ITS) for Foreign Language (FL). In: *Proceedings CALICO 2005: Call and the year of languages critical needs*. East Lansing: Michigan State University.

Finlay, A., Dix, J. (1996). An Introduction to Artificial Intelligence. London: UCL Press.

Foth, K., Hamerich, S., Schröder, I., Schulz, M., By, T. (2003). [X]CDG User guide. Version 1.3. Universität Hamburg.

Foth, K., Menzel, W., Schröder, I. (2005). Robust Parsing with Weighted Constraints. In: *Natural Language Engineering,* 11(1), S. 1-25.

Foth, K. (2005, 2006). Eine umfassende Contraint-Dependenz-Grammatik des Deutschen. Universität Hamburg.

Foth, K., Köhn, A., Beuck, N., Menzel, W. (2014). Because Size Does Matter: The Hamburg Dependency Treebank. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014).* Reykjavik.

Fraser, N. M. (1993). Dependency Parsing. Doktorarbeit, University of London.

Fraser, N. M. (1994). Dependency grammar. In: R. E. Asher (Hrsg.) *The Encyclopedia of Language and Linguistics, vol. 2,* S. 860–864.

Gal, A., Lapalme, G., Saint-Dizier, P., Somers, H. (1991). PROLOG for Natural Language Processing. Chinchester: John Wiley & Sons Ldt.

Gamon, M., Ringger, E., Zhang, Z., Moore, R., Corston-Oliver, S. (2002). Extraposition: a case study in German sentence realization. In: *Proceedings of COLING 2002*, Morristown.

Gamon, M., Leacock, C., Brockett, C., Dolan, W. B., Gao, J., Belenko, D., Klementiev, A. (2009). Using statistical techniques and Web search to correct ESL errors. In: *CALICO Journal*, 26(3), S. 491–511.

Garrett, N. (1987). A Psycholinguistic Perspective on Grammar and CALL. In: W. F. Smith (Hrsg.) *Modern Media in Foreign Language Education: Theory and Implementation*. Lincolnwood: National Textbook, S. 169-196.

Gasper, L., Sprenger, S. (1992). Technische und Theoretische Informatik. (1. Auflage). München: Bayerischer Schulbuch-Verlag.

Gass, S. M., Selinker, L. (2001). Second Language Acquisition. An Introductory Course. Mahwah: Erlbaum Associates.

Gazdar, G., Mellish, C. (1989). Natural Language Processing in Prolog – An Introduction to Computational Linguistics. Workingham: Addison-Wesley Publishing Company.

Granger, S., Vandeventer, A., Hamel, M. J. (2001). Analyse des corpus d'apprenants pour l'ELAO basé sur le TAL. In : *Traitement Automatique des Langues*, 42(2), S. 609-621.

Granger, S. (2002). A bird's eye view of learner corpus research. In: S. Granger, J. Hung, S. Petch-Tyson (Hrsg.) *Computer Learner Corpora, Second Language Acquisition and Foreign Language Teaching, Language Learning and Language Teaching 6*. Amsterdam & Philadelphia: Benjamins, S. 3-33.

Haapalainen, M., Majorin, A. (1995). GERTWOL und Morphologische Disambiguierung für das Deutsche. In: *Proceedings of the 10th Nordic Conference of Computational Linguistics*, University of Helsinki, Department of General Linguistics.

Haider, H. (1996). Downright down to the right. In: *Linguistik Aktuell*, 11, S. 245-271.

Hays, D. G., Ziehe, T. W. (1960). Studies in Machine Translation – 10: Russian Sentence- Structure Determination. Research Memorandum RM-2538. Santa Monica: The RAND Corporation.

Hays, D. G. (1964). Dependency theory: a formalism and some observations. In: *Language,* 40, S. 511– 525.

Hays, D. G. (1966). Parsing. In: David G. Hays (Hrsg.) *Readings in Automatic Language Processing*. New York: American Elsevier.

Handke, J. (1989). Computer Assisted Language Learning und Künstliche Intelligenz. In: *Die Neueren Sprache*, 88(1), S. 21-32.

Heift, T. (1998). Designed Intelligence: A Language Teacher Model. Doktorarbeit, Simon Fraser University, Burnaby.

Heift, T., Nicholson, D. (2000a). Theoretical and Practical Considerations for Web-based intelligent Language Tutoring Systems. In: G. Gauthier, C. Frasson, K. VanLehn (Hrgs.) *Intelligent Tutoring Systems. 5th International Conference, IITS 200, Montreal, Canada, June 19-23, 2000. Proceedings*. Montreal: Springer Verlag, S. 354-362.

Heift, T., Nicholson, D. (2000b). Enhanced Server Logs for Intelligent, Adaptive Web-Based Systems. In: *Intelligent Tutoring Systems. Proceedings of the Fifth International Conference, ITS 2000*, S.354-362.

Heift, T. (2001). Error-Specific and Individualized Feedback in a Web-Based Language Tutoring System: Do They Read It?. In: *ReCALL*, 13(2), S. 129-142.

Heift, T. (2002). Learner control and error correction in ICALL: browsers, peekers, and adamants. In: *CALICO Journal,* 19(2), S. 295–313.

Heift, T. (2003). Multiple Learner Errors and Feedback: A Challenge for ICALL Systems. In: *CALICO Journal*, 20(3), S. 549-560.

Heift, T. (2004). Corrective Feedback and Learner Uptake in CALL. In: *ReCALL*, 17(1), S. 32-46.

Heift, T., Schulze, M. (2007). Errors and Intelligence Computer-Assisted Language Learning: Parsers and Pedagogues. New York: Taylor & Francis Group.

Heift, T. (2010). Developing an Intelligent Language Tutor. In: *CALICO Journal*, 27(3), S. 443-459.

Heinecke, J., Kunze, J., Menzel, W., Schröder, I. (1998). Eliminative Parsing with Graded Constraints. In: *Proceedings of ColingAL '98,* S. 526-530.

Hendrickson, J. M. (1979). Evaluating Spontaneous Communication Through Systematic Error Analysis. In: *Foreign Language Annals*, 12(5), S. 357-364.

Hergenhahn, B. R., Olson, M. (1997). An introduction to theories of learning. London: Prentice-Hall International.

Holland, V., Maisano, R., Alderks, C., Martin, J. (1993). Parsers in Tutors: What Are They Good For?. In: *CALICO Journal*, 11(1), S. 28-46.

Holland, V., Kaplan, J., Sams, M. (1995). Intelligent Language Tutors: Theory Shaping Technology. New Jersey: Lawrence Erlbaum Associates Inc.

Hubbard, P. (2009). A general introduction to Computer-assisted Language Learning. In: P. Hubbard (Hrsg.) *Computer-Assisted Language Learning: Critical Concepts in Linguistics*. New York: Routledge, S. 1-20.

James, C. (1998). Errors in Language Learning and Use. Exploring Error Analysis. London: Longman.

Jodai, H. (2012). Error Analysis in a New Context: A Comparative Study of Errors in General Linguistics and CALL Author. In: *The Iranian EFL Journal*, 8(6), S. 332-345.

Johns, T. (1994). From printout to handout: grammar and vocabulary teaching in the context of data-driven learning. In: T. Odlin (Hrsg.) *Perspectives on Pedagogical Grammar.* Cambridge: Cambridge University Press, S. 293–313.

Johnson, M. (1988). Attribute-Value Logic and the Theory of Grammar. Lecture Notes n. 16. CSLI: Stanford University.

Johnston, I. (1988). Gibber. Doktorarbeit. Dundee: University of Dundee.

Jung, U. (2005). CALL: past, present and future – a bibliometric approach. In: *ReCALL*, 17(01), S. 4-17.

Juozulynas, V. (1994). Errors in the Composition of Second-Year German Students: An Empirical Study of Parser-Based ICALI. In: *CALICO Journal*, 12(1), S. 5-17.

Jurafsky, D., Martin, J. (2006). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.

New Jersey: Prentice Hall.

Keshavarz, M. H. (2011). Contrastive Analysis & Error Analysis. Tehran: Rahnama Press.

Kleene, S. C. (1956). Representation of Events in Nerve Nets and Finite Automata. In: C. E. Shannon, J. McCarthy, W. R. Ashby (Hrsg.) *Automata Sudies.* Princeton: Princeton University Press.

Kohn, K. (1994). Distributive Language Learning in Computer-Based Multilingual Communication Environment. In: H. Jung, R. Vanderplank (Hrsg.) *Barriers and Bridges: Media Technology in Language Learning. Proceedings of the 1993 CETaLL Symposiu, on the Occasion of the 10th AILA World Congress in Amsterdam.* Frankfurt am Main: Peter Lang, S. 31-43.

Krashen, S. D. (1985). The Input Hypothesis. Issues and implications. New York: Longman.

Kreindler, I. (1998). Designing Feedback that is Hard to Ignore: A Boost from Multimedia. In: K. Cameron (Hrsg.) *Multimedia CALL: Theory and Practice*. Exeter: Elm Bank Publications, S. 243-250.

Kübler, S., Prokic, J. (2006). Why is German Dependency Parsing More Reliable than Costituent Parsing?. In: *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories (TLT)*. S. 7-18.

Kübler, S., McDonald, R., Nivre, J. (2009a). Dependency Parsing. In: *Synthesis Lectures on Human Language Technologies 2*, S. 1-127.

Kübler, S., Rehbein, I., van Genabith, J. (2009b). A Testsuite for Testing Parser Performance on Complex German Grammatical Constructions. http://publikationen.ub.uni-frankfurt.de/opus4/frontdoor/index/index/docId/11889 <zuletzt abgerufen am 04.06.2015>

Kulik, J. A., Kulik, C.-L. C. (1988). Timing of Feedback and Verbal Learning. In: *Review of Educational Research,* 58(1), S. 79-97.

Lado, R. (1957). Linguistics Across Cultures: Applied Linguistics for Language Teachers. Ann Arbor: The University of Michigan Press.

Lafferty, J. D., McCallum, A., Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *Proceedings of the Eighteenth International Conference on Machine Learning,* San Francisco: Morgan Kaufmann Publishers Inc, S. 282–289.

Lee, M. J. (2003). Error Analysis of Written Texts by Learners of German as a Foreign Language. Masterarbeit. Waterloo: University of Waterloo.

Lennon, P. (1991). Error: some problems of definition, identification, and distinction.

In: *Applied Linguistics 12(2),* S. 180–196.

Leontiev, A. A. (1981). Psychology and the language learning process. Oxford: Pergamon Press.

Levin, L. (1993). Translators Workstations and Language Learners' Workstations. In: S. Nirenburg (Hrgs.) *Progress in Machine Translation*. Amsterdam: IOS Press, S. 287-290.

Levy, M. (1997). Computer-Assisted Language Learning. Context and Conceptualisation. Oxford: Clarendon Press.

Levy, M., Hubbard, P. (2005). Why call CALL CALL?. In: *Computer Assisted Language Learning,* 18(3), S. 143–149.

Lieberman, D. (1990). Learning. Behavior and cognition. Belmont, CA: Wadsworth Publishing Company.

Lightbown, P. M., Spada, N. (2006). How languages are learned (3. Auflage). Oxford: Oxford University Press.

Long, M. (1996). The role of the linguistic environment in second language acquisition. In: W. C. Ritchie, T. K. Bhatia (Hrsg.) *Handbook of second language acquisition*. San Diego: Academic Press.

Loritz, D. (1992). Generalized Transition Network Parsing for Language Study: The GPARS System for English, Russian, Japanese and Chinese. In: *CALICO Journal,* 10(1), S. 5-22.

Lyster, R., Ranta, L. (1997). Corrective feedback and learner uptake. In: *Studies in Second Language Acquisition*, 19, S. 37-66.

Mangenot, F. (1997). Synthèse de trois cours de FLE sur CD-ROM. In: *Les Cahiers de l'ASDIFLE N°9, Multimédia et français langue étrangère, Actes des 19e et 20e rencontres*. Paris: Association de Didactique du Français Langue Etrangère, S. 79-88.

Marino, A. (2000/01). Automatische Fehleranalyse der deutschen Sprache. Ein Verfahrenansatz. Masterarbeit. Universität Ca' Foscari, Venedig.

Mates, B. (1961). Stoic Logic. Berkeley: University of California Press.

Matthews, C. (1992a). Intelligent CALL (ICALL) Bibliography. Hull: CTI Centre for Modern Languages.

Matthews, C. (1992b). Fundamental Questions in ICALL. In: J. Thompson, C. Zaehner (Hrsg.) *Proceedings of the ICALL Workshop, UMIST, September 1991*. Hull: University of Hull, CTI Centre for Modern Languages, S. 77-89.

Matthews, C. (1992c). Going AI. Foundations of ICALL. In: *Computer Assisted*

Language Learning, 5(1-2), S. 13-31.

Matthews, C. (1993). Grammar Frameworks in Intelligent CALL. *CALICO Journal,* 11(1), S. 5-27.

Matthews, C. (1994). Intelligent Computer Assisted Language Learning as Cognitive Science: The Choice of Syntactic Frameworks for Language Tutoring. In: *Journal of Artificial Intelligence in Education,* 5(4), S. 533-556.

Mel'čuk, I. A. (1988). Dependency Syntax: Theory and Practice. New York: State University of New York.

Menyuk, P. H. (1971). The acquisition and development of language. Englewood Cliffs, New Jersey: Prentice-Hall Inc.

Menzel, W., Schröder, I. (1999). Error Diagnosis for Language Learning Systems. In: M. Schulze, M.-J. Hamel, J. Thompson (Hrsg.) *Language Processing in CALL. ReCALL Special Publication (Proceedings of a One-Day Conference "Natural Language Processing in Computer-Assisted Language Learning" Held at UMIST, May 9, 1998, Organised be the Centre of Computational Linguistics, UMIST, in Association with Eurocall.* Hull: CTICML, S. 20-30.

Menzel, W. (2004). Errors, Intentions, and Explanations: Feedback Generation for Language Tutoring Systems. In: *InSTIL/ICALI 2004*, Venedig.

Meurers, D. (2012). Natural Language Processing and Language Learning. In: C. A. Chapelle (Hrsg.) *The Encyclopedia of Applied Linguistics*. Blackwell Publishing Ltd.

Müller, S. (1994). Prolog und Computerlinguistik. Teil 1 – Syntax. https://hpsg.fu-berlin.de/~stefan/PS/prolog.pdf <zuletzt abgerufen am 04.06.2015>

Müller, S. (2002). Complex Predicates: Verbal Complexes, Resultative Constructions, and Particle Verbs in German. www.researchgate.net/publication/258332361_Complex_Predicates_Verbal_Complexes_Resultative_Constructions_and_Particle_Verbs_in_German/file/5046352af3d39af247.pdf <zuletzt abgerufen am 04.06.2015>

Müller, S. (2005). Zur Analyse der deutschen Satzstruktur. In: *Linguistische Berichte*. http://www.linguistik.hu-berlin.de/institut/professuren/korpuslinguistik/events/lesezirkel-gbg/satz-lb.pdf <zuletzt abgerufen am 04.06.2015>

Müller, G. (2006). On extraposition and successive cyclicity. In: Syntax: Critical Concepts in Linguistics, vol. III. Routledge.

Müller, S. (2013). Grammatiktheorie. Institut für Deutsche und Niederländische Philologie, Freie Universität Berlin. http://hpsg.fu-berlin.de/~stefan/Pub/grammatiktheorie.html <zuletzt abgerufen am 04.06.2015>

Murray, J. H., Morgenstern, D., Furstenberg, G. (1989). The Athena Language Learning Project: design issues for the next generation of computer-based language learning tools. In: W. F. Smith (Hrsg.) *Modern Technology in Foreign Language Education.* Lincolnwood: National Textbook, S. 97–118.

Nagata, N. (1993). Intelligent computer feedback for second language instruction. In: *The Modern Language Journal*, 77(3), S. 330-339.

Nagata, N., Swisher, M. V. (1995). A Study of Consciousness-Raising by Computer: The Effect of Metalinguistic Feedback on Second Language Learning. In: *Foreign Language Annals*, 28(3), S. 337-347.

Nagata, N. (1996). Computer vs. Workbook Instruction in Second Language Acquisition. In: *CALICO Journal*, 14(1), S. 53-75.

Nagata, N. (2009). Robo-Sensei's NLP-based error detection and feedback generation. In: *CALICO Journa*l, 26(03), S. 562-579.

Nederhof, M.-J., Satta, G., (2010). Theory of Parsing. In: A. Clark, C. Fox, S. Lappin (Hrsg.) *The Handbook of Computational Linguistics and Natural Language Processing.* Blackwell Publishing Ltd, S. 105-130.

Nerbonne, J. (2003). Natural Language Processing in Computer-Assisted Language Learning. In: R. Mitkov (Hrgs.) *The Oxford Handbook of Computational Linguistics*. Oxford: Oxford University Press, S. 670-698.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., Yuret, D. (2007). The CoNLL 2007 shared task on dependency parsing. In: *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*. S. 915–932.

Nivre, J. (2009). Non-Projective Dependency Parsing in Expected Linear Time. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Suntec, Singapore. Association for Computational Linguistics, S. 351–359.

Papert, S. (1981). Jaillissement de l'ésprit : ordinateurs et apprentissage. Paris : Flammarion.

Patten, T. (1992). Computer and Natural Language Parsing. In: C. S. Butler (Hrsg.) *Computers and Written Texts.* Oxford: Blackwell, S. 29-52.

Pennington, M. (2004). Electronic media in second language writing: an overview of tools and research findings. In: S. Fotos, C. M. Browne (Hrsg.) *New Perspectives on CALL for Second Language Classrooms*. Mahwah: Lawrence Erlbaum, S. 69–92.

Pereira, F. C. N., Warren, D. H. D. (1980). Definite Clause Grammars for Language Analysis – A Survey of the Formalism and a Comparison with Augmented Transition Networks. In: *Artificial Intelligence,* 13, S. 231-278.

Pollard, C. J., Sag, I. A. (1987). Information-Based Syntax and Semantics. Chicago: University Press.

Pollard, C. J., Sag, I. A. (1994). Head-Driven Phrase Structure Grammar. Chicago: University Chicago Press.

Pressey, S. L. (1927). A machine for automatic teaching of drill material. In: *School and Society*, 25, S. 549-552.

Richards, J. C. (1971). A Non Contrastive Analysis Approach to Error Analysis. In: *English Language Teaching*, 25, (3), S. 204-214.

Richards, J. C. (1974). Error Analysis: Perspectives on Second Language Acquisition. London: Longman.

Rimrott, A., Heift, T. (2008). Evaluating automatic detection of misspellings in German. In: *Language Learning and Technology,* 12(3), S. 73–92.

Sag, I. A., Wasow, T., Bender, E. M. (2003). Syntactic Theory: A Formal Introduction (2. Auflage). Stanford: Center for the Study of Language and Information.

Salaberry, R. (1996). A Theoretical Foundation for the Development of Pedagogical Tasks in Computer-Mediated Communication. In: *CALICO Journal*, 14(1), S. 5-34.

Sanders, A. F., Sanders, R. H. (1989). Syntactic Parsing: A Survey. In: *Computer and the Humanities*, 23, S. 13-30.

Schiller, A., Teufel, S., Stockert, C. (1999). Guidelines für das Tagging deutscher Textcorpora mit STTS (Kleines und großes Tagset). Universität Stuttgart: Institut für maschinelle Sprachverarbeitung.

Schmid, H., Fitschen, A., Heid., H. (2004). A German computational morphology covering derivation, composition, and inflection. In: *Proceedings of the IVth International Conference on Language Resources and Evaluation (LREC 2004)*, S. 1263–1266.

Schmitz, U. (1992). Computerlinguistik. Opladen: Westdeutscher Verlag.

Schneider, G. (1998). A Linguistic Comparison of Constituency, Dependency and Link Grammar. Lizentiatsarbeit, Universität Zürich.

Schneider, G. (2008). Hybrid Long-Distance Functional Dependency Parsing. Doktorarbeit, Institute of Computational Linguistics, University of Zürich.

Schneiderman, B. (1992). Designing the user interface. Strategies for effective human-computer interaction. MA: Addison-Wesley Publishing Company.

Schulze, M. (2003). Grammatical Errors and Feedback: Some Theoretical Insights. In:

CALICO Journal, 20(3), S. 437-450.

Schwind, C. (1988). Sensitive parsing: error analysis and explanation in an intelligent language tutoring system. In: *Proceedings of the 12th conference on Computational linguistics*. S. 608-613.

Selinker, L. (1974). Interlanguage. In: J. C. Richards (Hrsg.) *Error Analysis: Perspectives on Second Language Acquisition.* London: Longman, S. 31-54.

Selinker, L. (1992). Rediscovering Interlanguage. London: Longman.

Selva, T. (2002). Génération automatique d'exercices contextuels de vocabulaire. *TALN 2002*, S. 185-194.

Sennrich, R., Schneider, G., Volk, M., Warin, M. (2009). A New Hybrid Dependency Parser for German. In: *Proceedings of the German Society for Computational Linguistics and Language Technology* 2009. Potsdam, S. 115–124.

Sennrich, R., Volk, M., Schneider, G. (2013). Exploiting Synergies Between Open Resources for German Dependency Parsing, POS-tagging, and Morphological Analysis. In: *Proceedings of Recent Advances in Natural Language Processing 2013*. Hissar, S. 601-609.

Sennrich, R., Kunz, B. (2013). Zmorge: A German Morphological Lexicon Extracted from Wikitionary. In: *The LREC 2014 Proceedings of the Ninth International Conference on Language Resources and Evaluation,* Reykjavik, S. 1063-1067.

Sgall, P. (1994). Dependency-based formal description of language. In: R. E. Asher (Hrsg.) *The Encyclopedia of Language and Linguistics*, vol. 2. Oxford: Pergamon Press, S. 867–872.

Sharples, M. (1989). Computer and Thought: A Practical Introduction to Artificial Intelligence. Cambridge: MIT Press.

Skinner, F. (1968). La révolution scientifique de l'enseignement. Bruxelles: Charles Dessart.

Sutcliffe, A. (1995). Human-computer interface design. London: Macmillan Press Ltd.

Swartz, M., Yazdani, M. (1992). Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication. Berlin: Springer Verlag.

Tanenhaus, M., Spivey-Knowlton, M. (1995). Integration of visual and linguistic information in spoken language comprehension. In: *Science*, 298, S. 1632-1634.

Taylor, H. S. (1998). Computer Assisted Text Production: Feedback on Grammatical Errors Made by Learners of English as a Foreign Language. Masterarbeit. Manchester: University of Manchester.

Tesnière, L. (1959). Elements de syntaxe structurale. Paris : Klincksieck.

Tezarede, C. (1999/00). Probleme und neuere Perspektiven im Unterricht DaF in Italien. Eine Studie. Thesis. Universtità Ca' Foscari di Venezia.

Thorndike, E. L. (1913). Educational Psychology, Vol.2: The Psychology of Learning. New York: Teachers College.

Tolman, E. C. (1932). Purpose Behavior in Animals and Men. New York: Appleton-Century-Crofts.

Tribble, C., Barlow, M. (2001). Using Corpora in Language Teaching and Learning. *Special issue of Language Learning and Technology*, 5(03).

Tschichold, C. (1999). Grammar checking for CALL: Strategies for improving foreign language grammar checkers. In: K. Cameron (Hrgs.) *CALL: Media, design and applications*. Exton: Swets & Zeitlinger, S. 203-222.

Upshall, M. (Hrsg.) (1993). The Hutchinson concise encyclopedia (Updated 1994 Edition). Oxford: Helicon Publishing Ltd.

van der Linden, E. (1993). Does Feedback Enhance Computer-Assisted Language Learning?. In: *Computers and Education,* 21(1-2), S. 61-65.

Versley, Y. (2005). Parser evaluation across text types. In: *Fourth Workshop on Treebanks and Linguistic Theories (TLT)*. Barcelona.

von Wittich, B. (1967). Error Analysis: Implications for the Teaching for Foreign Languages. In: *Foreign Language Annals*, 12(4), S. 315-317.

Wagner, J., Foster, J., van Genabith, J. (2007). A Comparative Evaluation of Deep and Shallow Approaches to the Automatic Detection of Common Grammatical Errors. In: *EMNLP-CoNLL 2007 - Joint Meeting of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning, 28-30 June 2007*, Prague, Czech Republic.

Warschauer, M. (1996). Computer-Assisted Language Learning. An Introduction. In: S. Fotos (Hrsg.) *Multimedia Language Teaching*. Tokyo: Logos Publisher, S. 3-20.

Weischedel, R. M., Sondheimer, N. K. (1983). Meta-Rules as a Basis for Processing Ill-Formed Input. In: *American Journal of Computational Linguistics,* 9(3-4), S. 161-177.

Winograd, T. (1983). Language as a Cognitive Process: Syntax. Kapitel 5. In: *Augmented Transition Networks*, Addison-Wesley: Reading, MA.

Woods, W. A. (1970). Transition network grammars for natural language analysis. In: *Communication of the ACM*, 13(10), S. 591-606.

Woods, W. A. (1983). What's Important About Knowledge Representation?. In: *Computer*, 16(10), S. 22-27.

Yazdani, M., Uren, J. (1988). Generalising Language-Tutoring Systems: A French/Spanish Case Study, Using LINGER. In: *Instructional Science,* 17, S. 179-188.

Yoshii, M. (2006). L1 and L2 glosses: their effects on incidental vocabulary learning. In: *Language Learning & Technology*, 10(3), S. 85–101.

Zielinski, A., Simon, C. (2009). Morphisto – an open source morphological analyzer for German. In: *Proceedings of the 2009 conference on Finite-State Methods and Natural Language Processing: Post-proceedings of the 7th International Workshop FSMNLP 2008*. Amsterdam: IOS Press, S. 224–231.