Evaluation von State of the Art Syntaxparsern auf deutschen Romanen des 16. bis 20. Jahrhunderts

Bachelorarbeit

Vorgelegt am Lehrstuhl für Künstliche Intelligenz und Angewandte Informatik der Universität Würzburg bei Professor Dr. Puppe

Von: Julian Tritscher
Betreuer: Markus Krug
Würzburg, 19.02.2016
Mat.: 1920865

Inhaltsverzeichnis

1 Abstract

Goal of this bachelor thesis is the evaluation of results gained from processing sentences of a foreign domain with several german state-of-the-art syntax parsers. While this paper contains a general overview of important syntax parser terminology, as well as an introduction to common syntax parser architecture, the main focus lies on the experiment of finding the best running parser on the given domain, in addition to gaining an impression on the most common sources of errors to consider when choosing a parser for a specific domain. Our findings include issues with long, complex sentences for all parsers, as well as problems being caused by ancient or otherwise non-standard words. While all parsers show significantly lower scores in comparison to existing evaluations on fitting domains, the ParZu dependency parser shows quite substantially better results then all other parsers.

2 Einleitung

Das automatische Analysieren von Sätzen nach Syntax und Semantik ist ein weit reichendes, aktuelles Forschungsgebiet mit vielen unterschiedlichen Ansätzen. Während auf der einen Seite neue Technologien für die Nutzung des sog. „Parsens“ erschlossen werden [19], finden gleichzeitig Wettbewerbe zur Messung dieser Ansätze statt [17]. Diese Wettbewerbe der „Shared Tasks“ der CoNLL Konferenz für Natürliches Sprach Lernen(Conference on Natural Language Learning) [11] beschäftigten sich ins besondere mit der Ermittlung der effektivsten Methoden im Bereich des Syntax Parsens. Die Aufgabe des Syntax Parsers besteht in der satzweisen Analyse eines Dokuments mittels der Unterteilung eines Satzes in seine einzelnen syntaktischen Bestandteile. Erreicht wird dies durch vorrangehendes Training des Parsers mit Hilfe einer vorhandenen Menge an bereits korrekt markierten Sätzen, den sog. „Treebanks“, durch Maschinenlernen oder manuelles Erstellen einer Regelmenge. Für standardmäßige Messung verschiedener Parser werden hier beim Training die einzelnen Parser bestmöglich an die zu untersuchende Textart, die sog. „Domäne“ angepasst. Dies erfordert allerdings genaueres Wissen um die Beschaffenheit der anschließend zu untersuchenden Texte, und ist demnach nicht immer möglich und/oder sinnvoll. Dieser Problematik entsprechend sollen in diesem Paper fünf moderne Parser auf ihre Leistungsfähigkeit in einer ihnen unbekannten Domäne untersucht werden. Die Arbeit soll Aufschluss geben welcher Parser die wenigsten Schwierigkeiten auf der unbekannten Domäne zeigt. Außerdem sollen die einzelnen Fehler der Parser per Hand untersucht werden, um so einen Eindruck für die Fehlerquellen beim Parsen auf unbekannten Domänen zu erhalten.

Zunächst soll ein Überblick über die generellen Bestandteile eines Parsers gegeben werden. Darüber hinaus werden die beiden am weitest verbreiteten Ansätze des Konstituenz und Dependenz Parsens vorgestellt, und einige Umsetzungen dieser Strukturen erläutert. Auch soll ein Einblick in die Evaluation der Leistung verschiedener Parser durch Vorstellung eines „shared tasks“ der Konferenz für natürliches Sprachlernen (Conference on Natural Language Learning CoNLL) gegeben werden. Anschließend werden die fünf in unserer Arbeit verwendeten Parser betrachtet und gegebene Besonderheiten in ihrer Funktionsweise analysiert. Es folgt ein Überblick über Aufbau und Ablauf unseres Experiments zur Ermittlung des effizientesten Parsers. Insbesondere wird hier unsere Testmenge, sowie deren Beschaffenheit vorgestellt. Das Anschließen der einzelnen Parser wird beschrieben, eben so wie die Erstellung unserer Musterlösung, welche für die Messung der Leistungsfähigkeit der einzelnen Parser benötigt wird. Des Weiteren wird erläutert, auf welcher Ebene die einzelnen Parser Ergebnisse mit der Musterlösung verglichen werden. Die Art der Auswertung der Ergebnisse der einzelnen Parser wird ebenfalls beschreiben. Schließlich werden die Resultate der Parser auf verschiedenen Ebenen betrachtet und die Fehler der einzelnen Parser manuell analysiert. Dies ermöglicht einen Einblick über die Leistungsfähigkeit der Parser, sowie einen Aufschluss über zu berücksichtigende Faktoren bei der Wahl eines Syntax Parsers für eine gegebene Domäne.

3 Theorie des Syntax Parsens

Im diesem Kapitel soll zunächst ein Überblick über die grundlegenden Begriffe des Parsens geschaffen werden. Zusätzlich werden einige bekannte Ansätze erläutert und ein Ausblick auf zukünftige Technologien gegeben. Auch auf die Ermittlung des State of the Art für Syntax Parser soll kurz eingegangen werden.

3.1 Tagger

Die Grundlage unserer syntaktischen Satzanalyse bildet ein sogenannter Part-of-Speech Tagger. Dieser Tagger ordnet jedem Wort innerhalb eines Satzes ein Part-of-Speech Tag zu, welches die Wortart des Wortes im Satz beschreibt.

Der für unsere Forschungsarbeit verwendete Tagger, Helmut Schmids „Tree-Tagger“ [15], bestimmt seine Part-of-Speech Tags durch den Vergleich der eingegebenen Sätze mit einem privaten, bisher unveröffentlichten Trainingscorpus in folgender Weise:

Der Tagger besitzt ein Lexikon, in welchem alle ihm durch den Trainingscorpus bekannten Wörter mit ihren Tags und deren Vorkommenswahrscheinlichkeiten aufgeführt werden. Dies wird als lexikale Wahrscheinlichkeit bezeichnet. Auch besitzt das Lexikon Einträge über häufige Verknüpfungen von Worten, welche zur Erkennung von Phrasen genutzt werden können. Dies stellt die kontextuelle Wahrscheinlichkeit eines Tags dar. Ein sog. „Hidden Markov Model“ analysiert nun einen Eingabesatz unter Berücksichtigung beider Wahrscheinlich-

keiten des Lexikons. Die erweiterte Erklärung der Vorgehensweise eines Part-of-Speech Taggers ist nachschlagbar bei Clark et. Al. [4] S. 192ff.

So entsteht für jeden Satz eine abstrakte Satzstruktur(Abbildung 1), was den Parsern ermöglicht, bei unbekannten Wörtern auf die Part-of-Speech Tags zurückzugreifen.

Während einige Parser die Part-of-Speech Tags als zusätzliche Eingabe zu den zu parsenden Sätzen erwarten, besitzen andere Parser ihre eigenen Tagger Methoden, welche zusammen mit dem Hauptprogramm ausgeführt werden.

NE	VVFIN	ART	NN	-PUNCT-
Lydia	bewegte	die	Lippen	.

Abbildung 1: Hier wird ein Satz mit zugehoerigen Part-of-Speech Tags dargestellt.

3.2 Morphology

[1] Zusätzlich zum Setzen von Part-of-Speech Tags können Wörter auch mit Morphology Informationen versehen werden. Diese werden durch das Aufspalten des Wortes in Präfix, Wortstamm und Suffix, und deren Abgleichen mit speziellen Regeln oder Lexika ermittelt. Eine morphologische Analyse besteht aus einer genaueren Analyse des Wortes nach Genus, Kasus, Numerus und Person, und versorgt einen Parser somit mit zusätzlichen Informationen, welche insbesondere das Bestimmen der Art einer Phrase, sowie das Finden der generellen syntaktischen Struktur des Satzes erleichtert.

Auch hier existieren Parser, die ihre eigenen Algorithmen zur morphologischen Analyse verwenden. Ein Zuführen der Morphology Tags mit der Eingabe des Textes ist allerdings auch üblich.

3.3 Syntax Parser

Syntax Parser sind Programme, welche, mit Hilfe von Regeln oder durch eigenes Lernen an vorbearbeiteten Beispielen, Sätze auf ihre Satzteile und deren syntaktische Funktion automatisch analysieren. Während Parser existieren, welche ausschließlich die Struktur eines Satzes betrachten, vergeben andere Parser, wie Tagger, Labels/Tags, die eine tiefere Analyse des Satzes, z.B. durch die Unterscheidung von Subjekt und Akkusativobjekt, ermöglichen. Doch auch der generelle Ansatz, sowie die Funktionsweise der Parser können unterschiedlich sein. Die zwei am häufigsten verwendeten Grundarchitekturen für Syntax Parser, sowie einige übliche Umsetzungen, sollen im Folgenden kurz vorgestellt werden.

[1] Dies und das Folgende nach [4] S. 366ff

3.3.1 Dependenzgrammatik

[2] Die Dependenzgrammatik definiert die strukturelle Basis eines Dependenz Parsers und basiert auf der Annahme, dass die syntaktische Struktur eines Satzes durch binäre, asymmetrische Dependenzrelationen zwischen den Wörtern darstellbar ist. Jede Dependenzrelation besitzt ein zugehöriges Wort, den sog. „Head“ (Kopf) der Relation, sowie ein Bezugswort, den sog. „Dependent“. Auch besitzt jede Dependenzrelation ein Label, welches die Art der Beziehung zwischen beiden Worten, den sog. „Dependenz Typ“, beschreibt.

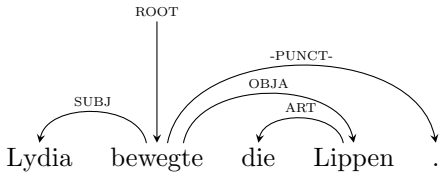

Abbildung 2: Hier wird ein Satz in Dependenz Grammatik dargestellt.

Am Beispiel aus Abbildung (2) erläutert, existiert eine syntaktische Verbindung zwischen den Worten „Lydia“ und „bewegte“, welche anzeigt, dass der Dependent, „Lydia“, in seiner Funktion als Subjekt („SUBJ“) in Verbindung zum Head, „bewegte“, steht.

Das zusätzlich eingefügte Wort „ROOT“ funktioniert als syntaktische Wurzel eines entstehenden Dependenzbaumes und beschreibt den Head des zentralen Verbs des Satzes.

Es besteht weiterhin die Grundannahme, dass für jeden grammatikalisch korrekten Satz ein zusammenhängender, den Satz vollständig syntaktisch beschreibender, Dependenzbaum erstellt werden kann.

Sämtliche Definitionen und Grundregeln für Dependenzbäume und deren Erstellung sind nachlesbar in Kübler et. Al. [10] Kap. 2.1.

3.3.2 Dependenz Parser

[3] Das Dependenz Parsen beschäftigt sich mit der Aufgabe, für eingegebene Sätze automatisch einen passenden Dependenzbaum zu finden. Für diese Problemstellung existieren zunächst verallgemeinert zwei unterschiedliche Ansätze, das sog. datengetriebene („data-driven“), und das grammatikbasierte („grammar based“) Dependenz Parsen, für die einige häufige Umsetzungen im Folgenden diskutiert werden sollen.

Datengetriebene Ansätze Datengetriebene Parseransätze greifen für die Erstellung von Dependenzbäumen auf maschinelles Lernen zurück. Für diese Parser wird also generell eine Trainingsmenge an Daten benötigt, die bereits

[2] Dies und das Folgende nach [10] Kap. 1.1
[3] Dies und das Folgende nach [10] Kap. 1.2

mit korrekt markierten Lösungen versehen ist und vom Parser für die Erstellung eines eigenen Modells verwendet werden kann. Auch hier existieren unterschiedliche Herangehensweisen, von denen die beiden häufig verwendeten Methoden des transitionsbasierten, sowie des graphbasierten Parsens in Folge betrachtet werden.

Transitonsbasiert [4] Ein transitionsbasierter Parser arbeitet über eine Zustandsmaschine, welche aus einer Menge von Zuständen, sowie einer Menge von Transitionen/Übergängen zwischen Zuständen besteht. Bei Eingabe eines Satzes in die Maschine wird nun Schritt für Schritt immer die wahrscheinlichste Transition ausgeführt und so der Zustand der Maschine verändert. Dies wird solange wiederholt, bis die Maschine einen terminalen Zustand erreicht, und somit einen Dependenzbaum zurückgibt.

Um allerdings festzustellen, welche Transition im momentanen Zustand der Maschine ausgeführt werden soll, werden sog. „Klassifikatoren" benötigt. Diese werden durch das Analysieren vormarkierter Beispielsätze, einem sog. Trainingscorpus, oder auch Treebank genannt, mit Hilfe von Maschinenlernalgorithmen einmalig ermittelt und als Modell für den Zugriff beim Parsen gespeichert.

Ein Parse-Algorithmus hat nun die Aufgabe, für den momentanen Zustand der Maschine mit Hilfe der Klassifikatoren des Modells die wahrscheinlichste Transition zu finden.

Graphbasiert [5] Graphbasierte Parser betrachten jedes Wort eines Satzes als Knoten und versuchen aus diesen einen zusammenhängenden, syntaktisch korrekten Dependenzbaum zu erstellen. Zur Gewichtung der verschiedenen Kanten werden wieder Klassifikatoren herangezogen, die durch Maschinenlernen im Modell des Parsers vorhanden sind. Da allerdings das Überprüfen sämtlicher Möglichkeiten für alle Knotenkonstellationen zu aufwendig wäre, liegt die Herausforderung dieses Parser Ansatzes in der Umsetzung eines effizienten Algorithmus. Ein Beispiel eines häufig verwendeten Parser Algorithmus wäre der gierige („greedy"), rekursive Chu-Liu-Edmonds Spannbaum Algorithmus, vorgestellt in Kübler et. Al. [10] S. 47.

Grammatik basierte Ansätze Während datengetriebene Parser auf ein durch Maschinenlernen trainiertes Modell zur Entscheidung des korrekten Dependenzbaums zurückgreifen, wird das Modell eines grammatikbasierten Parsers in Form einer Grammatik erstellt. Die Nichtterminale, Regeln, sowie deren Gewichtung sind von der Art der Grammatik abhängig. Im Folgenden sollen zwei unterschiedliche Ansätze des grammatikbasierten Parsens präsentiert werden.

Kontext-freie Dependenz Grammatik [6] Für den Ansatz der kontextfreien Dependenz Grammatik wird aus der Zugrunde liegenden Dependenz Gram-

[4] Dies und das Folgende nach [10] Kap. 3
[5] Dies und das Folgende nach [10] Kap. 4
[6] Dies und das Folgende nach [10] Kap.5.1

matik wie nach Abbildung (3) eine Kontext-freie Grammatik erstellt. Es werden als Nichtterminale der Grammatik die Heads der Relationen eingesetzt und diese mit ihren direkten Dependenten und sich selbst verknüpft. Nun können im Verlauf des Baumes Regeln in Form von Abbildung (3) betrachtet werden. Damit entsteht aus der Dependenz Struktur eine Kontext-freie Grammatik, auf die etablierte Parser Algorithmen für kontext-freie Grammatiken, wie z.B. der in Collins et. Al. [5] Kap. 3.4.2 aufgeführte CKY Algorithmus, angewendet werden können. Die Regeln der erstellten Grammatik, sowie deren Gewichtung können entweder durch Maschinenlernen aus einer Treebank gewonnen, oder per Hand erstellt und passend gewichtet werden.

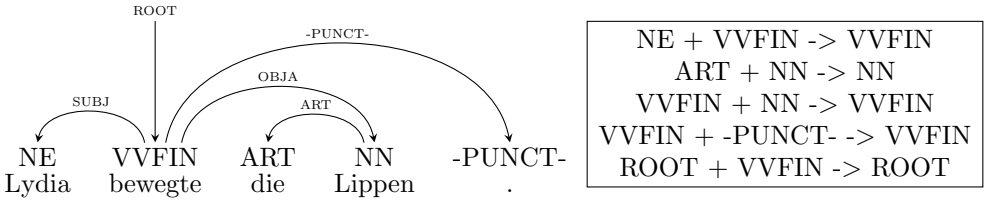

Abbildung 3: Hier wird die Umwandlung eines Satzes von Dependenz Struktur in eine Grammatik dargestellt.

Konstraintbasierte Dependenz Grammatik [7] Die Basis einer konstraintbasierten Dependenz Grammatik ist eine Menge von Einschränkungen. Diese Einschränkungen beziehen sich auf die Wortart des Wortes, also das Part-of-Speech Tag, und werden aus der Syntax der Sprache erstellt. Sie können gewichtet werden, basierend auf ihrer Vorkommenshäufigkeit. Wenn beispielsweise 97% aller deutschen Sätze ein Verb enthalten, dann kann eine Einschränkung erstellt werden, nach der ein Satz mit Gewichtung 0,97 ein Verb enthalten muss. Ein Parser Algorithmus hat hier also die Aufgabe, für einen Satz die Lösung auszugeben, welche insgesamt, unter Berücksichtigung der Gewichtungen, die wenigsten Einschränkungen verletzt. Auch können Einschränkungen auf die Abfolge von Worten bezogen werden. So könnte beispielsweise eine Einschränkung erstellt werden, bei der die Existenz eines Verbs im Satz mit 90% Wahrscheinlichkeit die Existenz eines Subjekts nach sich zieht. Einschränkungen selbst können hier wieder sowohl aus einer Treebank durch Maschinenlernen bezogen, als auch manuell geschrieben werden.

3.3.3 Konstituenz Grammatik

Im Gegensatz zu Dependenz basierten Strukturen, basiert die Konstituenz Grammatik auf einer Phrasenstruktur. Das Grundprinzip besteht darin, dass zunächst jedes Wort auf ein Nichtterminal zusammengefasst wird, welches seine Wortart beschreibt. Anschließend werden jeweils zwei Nichtterminale zu einem, ei-

[7] Dies und das Folgende nach [10] Kap. 5.2

ne Phrase beschreibendem, Nichtterminal zusammengefasst. Dies wird solange wiederholt, bis die Wurzel des Baumes, das Nichtterminal „S“, welches den gesamten Satz umfasst, erreicht wird.

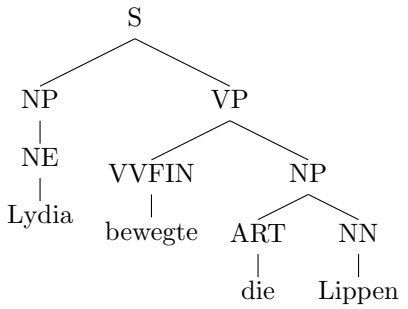

Abbildung 4: Hier wird ein Satz in Konstituenz Grammatik dargestellt.

Am Beispiel der Abbildung (4) wird also zunächst jedes Wort auf ein Nichtterminal seiner Wortart abgebildet. Anschließend wird das Nichtterminal des Wortes „die“ mit dem des Wortes „Lippen“ zu einer Nominalphrase „NP“ zusammengefasst, was eine Abhängigkeit beider Worte darstellt. Die Verknüpfung zwischen dieser „NP“ Phrase und dem Verb „bewegte“ wird mit als Verbphrase „VP“ markiert und schließlich mit der Nominalphrase, die aus dem Wort „Lydia“ besteht, zu einem vollständigen Satz „S“ zusammengefasst.

3.3.4 Konstituenz Parser

Das Konstituenz Parsen beschäftigt sich, wie in Abbildung(4) erkennbar, mit der Aufgabe, einen Satz automatisch in Phrasen zu unterteilen.

Auch hier bestehen, Dependenz Parsern entsprechend, verschiedene Ansätze mit der verallgemeinerten Unterteilung in datengetriebenes und grammatikbasiertes Konstituenz Parsen, auf die im Folgenden kurz eingegangen werden soll.

Datengetriebene Ansätze Während für Dependenz Parser reine datengetriebene Ansätze existieren, bestehen datengetriebene Konstituenz Parser meist aus Maschinenlernalgorithmen, verbunden mit einer passenden Konstituenz Grammatik. Diese Grammatik wird als Grundlage des Parsers gewählt, da die gesamte Struktur eines Konstituenz Baumes einfach in Grammatikform beschrieben werden kann.

Ein Beispiel für eine solche Hybrid Architektur bieten die in Clark et. Al. [4] Kap. 6.1-2 erörterte kombinatorische, kategorialsche Grammatik (eng. Combinatory categorial grammar „CCG“) mit konditionellem, log-linearem Parsing Modell. Während hier sämtliche möglichen Konstituenzbäume eines Satzes durch die CCG Grammatik bestimmt werden, übernimmt anschließend das log-lineare Parsing Modell die Bewertung der gefundenen Lösungen. Die Grammatik erstellt ihre Lösungsbäume durch die Kombination von Wörtern und/oder Phrasen zu

sog. „Features“ nach den ihr antrainierten Regeln. Das Modell betrachtet anschließend die einzelnen Features mit Hilfe von durch Maschinenlernen erstellten Häufigkeitsfunktionen, sowie Gewichten. Der datengetriebene Teil des Parsers ist hier also für die Gewichtung der möglichen Lösungsbäume, und somit für das Finden der wahrscheinlichsten Lösung zuständig.

Grammatikbasierte Ansätze Der grammatikbasierte Ansatz ist die generell am häufigsten gewählte Vorgehensweise von Konstituenz Parsern, da sich ein Konstituenzbaum recht unkompliziert als Grammatik ausdrücken lässt. Ein Beispiel für eine solche Architektur bietet die Umsetzung mittels PCFGs.

PCFGs [8] Die am weitesten verbreitete Vorgehensweise für das Konstituenz Parsen ist die Erstellung einer wahrscheinlichkeitsbasierten kontext-freien Grammatik (probabilistic context-free grammar „PCFG“). Diese basiert auf einer Regelmenge, in der Kombinationen von Worten und/oder Phrasen auf Phrasen abgebildet werden. Auch sind sämtliche Regeln nach ihrer Auftrittswahrscheinlichkeit gewichtet, was eine Suche nach dem wahrscheinlichsten Lösungsbaum ermöglicht. Beispielsweise ist der in Abbildung (5) gezeigte Konstituenzbaum die optimale von einem Parser gefundene Lösung, wenn sämtliche neben dem Baum abgebildeten Regeln zusammen die wahrscheinlichste Regelfolge für den Baum ergeben. Die Regelmenge, sowie deren Gewichtung können entweder von Hand, oder durch Maschinenlernen auf einer Treebank erstellt werden und stehen dem Parser Algorithmus anschließend in einem Modell zu Verfügung. Ein Beispielalgorithmus, welcher auf dieser Grammatik Konstituenzbäume erstellt, ist der CKY Algorithmus, beschrieben in Collins et. Al. [5] Kap. 3.4.2.

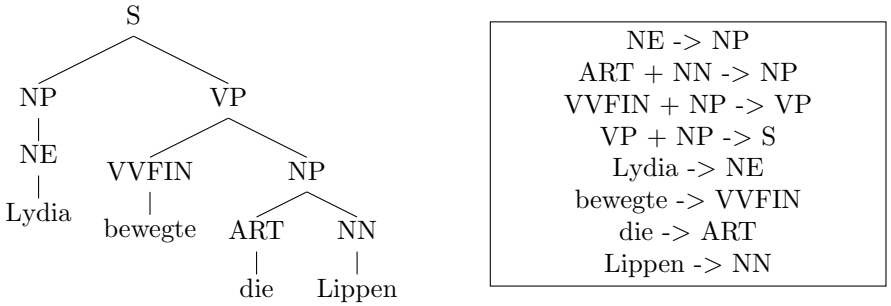

Abbildung 5: Hier wird die Umwandlung eines Satzes von Konstituenz Struktur in eine Grammatik dargestellt.

3.3.5 Gegenseitige Adaption von Ansätzen

Während Dependenz und Konstituenz Grammatiken sich von ihrem Aufbau grundsätzlich unterscheiden, gibt es doch Wege, die Struktur zumindest soweit

[8] Dies und das Folgende nach [5]

zu verändern, um sich der Ansätze der anderen Architektur bedienen zu können. Eine dieser Vorgehensweisen wurde bereits bei der Vorstellung der kontext-freien Dependenz Grammatik besprochen, wobei eine Dependenz Struktur abgeändert wurde, um auf ihr Algorithmen für die zuvor bei Konstituenz Parsern erwähnten PCFGs anzuwenden. Ein ähnlicher Ansatz in die andere Richtung wäre beispielsweise die Grundidee des Stanford Shift-Reduce Konstituenz Parsers nach Zhu et. Al. [23], welcher seine zu Grunde liegende Konstituenz Grammatik in eine Regelmenge einer Shift-Reduce Zustandsmaschine umwandelt und sich so das ebenfalls besprochene Konzept des transitionsbasierten Parsens für Dependenz Parser zu Nutze zu machen.

Es kann also beobachtet werden, dass beide Strukturen sich nicht gegenseitig ausschließen, und dem Entsprechend viele Parser einen gemischten Ansatz für ihre Architektur wählen, um möglichst viele Vorteile für ihre eigene Laufzeit und Treffsicherheit zu erhalten.

3.3.6 Ausblick auf zukünftige Architekturen: Neuronale Netze

[20] Neuronale Netze bezeichnen einen Ansatz der Informationsverarbeitung. Grundlage dieses Systems bietet, ähnlich dem menschlichen Gehirn, eine Menge an stark verknüpften Verarbeitungselementen, den sog. Neuronen. Diese beobachten den mit ihnen verknüpften Input und reagieren gegebenenfalls mit einem binären Signal. Wann Neuronen reagieren, und ob sie eine 0 oder 1 als Reaktion abgeben, kann den Neuronen durch Maschinenlernen antrainiert werden. Falls für den erhaltenen Input keine direkte Lösung antrainiert wurde, so wird durch spezielle Algorithmen der ähnlichste dem Neuron bekannte Input und dessen Lösung verwendet. Das gesamte neuronale Netz besteht aus vielen verschiedenen Neuronen, deren gesamte Ausgabe zur Ermittlung der benötigten Lösung der Informationsverarbeitung verwendert wird.

Im Bereich des Syntax Parsens bestehen bereits Ansätze, die neuronale Netzwerke für die Erstellung passender Syntax Bäume verwenden. Beispielsweise wird die Architektur eines rekursiven neuronalen Netzes bei Socher et. Al. [19] zur Erstellung einer Konstituenz Grammatik genutzt. Basiert wird die Architektur auf paarweiser Betrachtung jedes Wortes mit seinem linken und rechten Nachbarn. Jedes dieser Paare wird durch ein neuronales Netz betrachtet, welches den wahrscheinlichsten Parent-Knoten für das gegebene Paar findet. Sobald alle Nachbarnpaare untersucht sind, wird das Paar mit dem höchstwahrscheinlichsten Elternknoten aus der Menge der zu untersuchenden Paare entfernt und mit seinem Elternknoten ersetzt. Gleichzeitig wird eine Verbindung zwischen den beiden im Algorithmus entfernten Elementen zu ihrem Elternknoten im Konstituenzbaum erstellt. Dies wird so lange rekursiv wiederholt, bis ein vollständiger Konstituenzbaum ensteht. Trainiert wird das Netz, wie bei herkömmlichen daten-getriebenen Parsern, durch Maschinenlernen auf einer Treebank.

3.3.7 State-of-the-Art von Syntax Parsern: CoNLL Shared Tasks

[9] Die Konferenz für natürliches Sprachlernen (Conference on Natural Language Learning) ist eine jährlich von der Interessenvereinigung SIGNLL abgehaltene Versammlung, in der viele verschiedene Forschungsaspekte des Sprachlernens der Öffentlichkeit durch Vorträge, sowie die Erstellung eines umfassenden Sitzungsprotokolls vorgestellt werden.

Insbesondere beinhaltet die jährliche Konferenz eine gemeinsam zu bewältigende Aufgabe, den sog. „shared task“, für die Teams von Wissenschaftlern ihre Lösungsansätze präsentieren und deren Ergebnisse, sowie Funde im Sitzungsprotokoll der Konferenz festgehalten werden. Für diese Arbeit besonders relevant ist hier der shared task der CoNLL Konferenz 2007 [11], welcher sich mit der Evaluation verschiedenster Dependenz Parser auf mehreren Sprachen beschäftigt. Auch besitzt der Task eine weitere Aufgabe des Anpassens eines Parsers an eine unbekannte Domäne.

Mehrfachsprachen-Experiment Für die Evaluation der verschiedenen Parser auf untschiedlichen Sprachen wurden für jede Sprache ein festes Trainingsset mit mindestens 50.000 und höchstens 500.000 Tokens, sowie zu parsende Testdaten zwischen 4.500 und 7.500 Tokens zur Verfügung gestellt. Die Vorgehensweise des Parsers bei Training und Parsen war dagegen den Teilnehmern überlassen. Die gesamte Testmenge des Experiments beinhaltete insgesamt zwanzig verschiedene Parseransätze, die auf elf verschiedenen Sprachen getestet wurden. Es sind sowohl transitionsbasierte, als auch graphbasierte Parser aufzufinden. Die Resultate der Parser variieren allerdings für beide Parserarchitekturen. Ausgewertet wurden die Ergebnisse anhand der Anzahl der Dependenzrelationen mit Verweis auf das korrekte Head-Wort, dem sog. Labeled Attachment Score(LAS), definiert in Kübler et. Al. [10] Kap. 6.1.

Während Ansätze existieren, die auf allen Sprachen einen Durchschnittswert von 55% erreichen, zeigt der beste Parser einen Durchschnittswert von über 80% LAS. Auch für die unterschiedlichen Parserarchitekturen existieren starke Schwankungen, wobei kein klarer Favorit erkennbar ist. Der beste erreichte Labeled Attachment Score eines Parsers beträgt 89.61% für die englische Testmenge, bei der generell gute Ergebnisse zu erkennen sind. Griechisch auf der anderen Seite zeigt als syntaktisch kompliziertere Sprache mit 76.31% den niedrigsten Bestwert aller Sprachen auf.

Der mit einem Gesamtdurchschnitt von 80.32% am besten abschneidende Parser ist eine Version des auch in unserem Experiment verwendeten Malt Parsers, der allerdings durch spezielle Lern- und Parseralgorithmen an den Versuch des mehrsprachigen Parsens angepasst wurde und in Hall et. Al. [8] beschrieben wird.

Domänenadaptions-Experiment Im zweiten Experiment des CoNLL shared task 2007 wurde die automatische Adaption verschiedener Parser auf eine ih-

[9] Dies und das Folgende nach [17]

nen unbekannte Domäne betrachtet. Dies bedeutet, dass die zu parsenden Texte sich in Struktur, Komplexität, sowie Wortschatz von den für das Parsertraining verwendeten Mustern unterscheiden können. Der Aufbau dieses Experiments lautete, nach Dredze et. Al. [6] wie folgt: Der für das Experiment bereitgestellte Trainingskorpus der Parser bestand aus 15 tausend von per Hand annotierten englischen Zeitungsartikeln des Wall Street Journals. Die Testdomäne, welche den Teams zur Entwicklung des Parsers zur Verfügung stand, beinhaltete 200 annotierte englische Sätze aus dem Bereich der Biomedizin. Getestet wurden die Parser auf 200.000 englischsprachigen Sätzen ohne labels aus dem Fachbereich der Chemie, sowie auf englischen Eltern-Kind Dialogen. Ziel des Experiments war die Umsetzung eines Parsers, welcher unter Verwendung der annotierten Zeitungsartikel-Domäne für beide unbekannten Domänen treffsichere Ergebnisse liefern konnte.

Es soll an dieser Stelle kurz auf die beiden häufigsten Ansätze des Domänenadaptions-Experiments eingegangen werden. Weitere Ansätze sind in Nilsson et. Al. [11] Kap. 5.4.3, sowie in den dort aufgelisteten Papers zu finden.

Zum einen wurde versucht, das Modell des Parsers durch einen Feature-basierten Ansatz an eine unbekannte Domäne anzupassen. Dies beinhaltet das Bewerten der in der Trainingsdomäne vorkommenden Verknüpfungen von Worten, den sog. „Features“, nach der angenommenen Häufigkeit dieser Features in den zu parsenden Texten. Es sollen somit nur die Features der Trainingsmenge berücksichtigt werden, von denen angenommen wird, dass sie sich auf die Zieldomäne übertragen lassen. Diese überarbeitete Menge an Features kann anschließend zum Training eines neuen Parsers verwendet werden. Umsetzungen von Dredze et. Al. [6] zeigen hier allerdings bezogen auf das spezielle Experiment keine nennenswerten Verbesserungen.

Zum anderen wurde die Anpassung des Parsers über Ensemble-basierte Ansätze erforscht. Hier wurden die Trainingsdaten auf mehrere, unterschiedliche Parser aufgeteilt. Diese Parser wurden anschließend auf die Testmenge angewendet. Darauf wurden sämtliche Sätze, für die die Parser Ergebnisse glichen, zur ursprünglichen Testmenge hinzugefügt und schließlich ein einzelner Parser auf der gesamten Trainingsmenge, bestehend aus den Zeitungsartikel und den übereinstimmenden Sätzen, trainiert. Dieser Vorgang wurde von Dredze et. Al. [6], sowie Sagae et. Al. [14] untersucht.

Allgemein kann Folgendes über die Ergebnisse des Experiments gesagt werden: Während die Domäne der Eltern-Kind Dialoge durch eine zum Trainingssatz verschiedene Annotationsweise der Dependenzen mit einem Bestwert von nur 62.49% im Unlabeled Attachment Score (UAS) trotz der simplen Satzstruktur nur sehr niedrige Werte aufweist, zeigt die Domäne der chemischen Auszüge einige interessante Ergebnisse. Auch hier schwanken, wie im ersten Experiment, die Werte zwischen rund 80% und 50% im LAS. Zu beobachten ist allerdings, dass der Parser mit den besten Testergebnissen von 81.06% auf der angepassten Chemie Domäne im vorherigen Experiment ein Ergebnis von 89.01% im LAS für Englisch auf passender Domäne erreichte. Dies zeigt, dass die Treffsicherheit des Parsers sogar beim Versuch der automatischen Anpassung auf eine unbekannte Domäne deutlich abnimmt. Welche Probleme bei der automatischen syntakti-

schen Analyse einer unbekannten Domäne genau für solch eine Abnahme sorgen, soll anhand unseres Experiments des Parsens einer unbekannten Domäne ohne Anpassung der Parser untersucht werden.

4 Architektur verwendeter Parser

Das nächste Kapitel soll einen Einblick in die für unser Experiment verwendeten deutschen Syntax Parser bieten. Es soll ein kurzer Überblick über die Funktionsweise der Parser, die Struktur ihrer Ergebnisse, sowie vorhandene Besonderheiten gegeben werden.

4.1 Mate Parser

[10] Der für unsere Forschung verwendete Mate Parser ist ein transitionsbasierter Shift-Reduce Dependenz Parser, welcher sich einiger Zusatzfunktionen zur Verbesserung der Parser Genauigkeit bedient. Die Form der ausgegebenen Dependenzbäume des Parsers sind in Abbildung (6) zu erkennen. Trainiert ist Mate auf der Tiger Treebank, einem Dokument mit handmarkierten Sätzen im Umfang von 650.000 Wörtern, Satz- und Sonderzeichen, sogenannten „Tokens“. Die von Mate Parser verwendeten Dependenz Relations-Tags sind in Tabelle (8) und (9) im Anhang dieser Arbeit aufgeführt. Zusätzlich zur Parser Funktion, kommt Mate mit seinem eigenen Part-of-Speech Tagger, welcher zusammen mit den Hauptfunktionen ausgeführt wird.

Neben dem Modell für Tagger und Shift-Reduce Algorithmus, trainiert Mate, wie eine konstraintbasierte Dependenz Grammatik, zusätzlich ein Einschränkungslexikon, mit Hilfe dessen der Parser die Wahrscheinlichkeit der grammatikalischen Abhängigkeit von Worten bewerten, und so bessere Aussagen über die syntaktische Satzstruktur treffen kann.

Eine weitere Zusatzfunktion nutzt Mate mit der Verwendung von sog. „Wort Clustern“ im Einschränkungslexikon, welche passende Worte in Übergruppen zusammenfassen und so die Leistungsfähigkeit des Parsers bei Auftritt von seltenen oder unbekannten Wörtern verbessern.

Mit dieser Struktur erreichte die englische Version des Mate Parser in einer 2015 veröffentlichten Leistungsuntersuchung von Choi et. Al. [3] herausragende Ergebnisse.

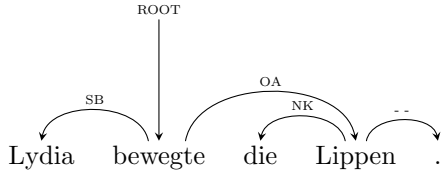

Abbildung 6: Hier wird ein Satz annotiert von Mate Parser dargestellt.

[10] Dies und das Folgende nach [1]

4.2 Malt Parser

[11] Malt Parser wird von seinen Entwicklern als „Parser Generator“ bezeichnet. Von der Grundlage eines transitionsbasierten Shift-Reduce Parsers aus, werden dem Benutzer sowohl vorprogrammierte Protokolle und Algorithmen, als auch die Möglichkeit der Umsetzung eines eigenen Lösungsweges zur Verfügung gestellt. Während die Art der Maschine durch die Shift-Reduce Architektur festgelegt ist, kann der Benutzer Eingaben, Lern- und Parser-Algorithmen, sowie die Klassifikatoren des Parsers selbst umsetzen, um so Malt bestmöglich an das benötigte Einsatzgebiet anzupassen. Zusatzfunktionen, sowie die ermöglichte Anpassung an eine gegebene Domäne wurden in unserem Experiment allerdings bewusst ungenutzt gelassen, da die Leistung des Parsers auf einer ihm unbekannten Domäne untersucht werden soll.

Da Malt Parser kein standardmäßiges deutsches Modell für seinen Parser liefert, wurde das für unser Experiment verwendete Modell auf der TüBa-D/Z Treebank der Universität Tübingen nach Standardeinstellungen des Parsers trainiert. Auch besitzt Malt Parser keinen integrierten Tagger und macht demnach sowohl von den von uns generierten Part-of-Speech Tags, als auch benötigten Morphology Tags Gebrauch, welche dem Parser zusammen mit den zu parsenden Sätzen im CoNLL Format als Eingabe übergeben werden. Die von Malt Parser verwendeten Dependenz Relations-Tags der TüBa-D/Z Treebank sind in Tabelle (8) und (9) im Anhang dieser Arbeit aufgeführt. Die Ausgabe des TüBa-D/Z Models des Parsers ist in Abbildung (7) dargestellt.

Malt Parser erziehlte in der Vergangenheit mit frühen Iterationen seines englischen Parsers herausragende Parsing Ergebnisse, beispielsweise im bereits erwähnten 2007er CoNLL shared task on Dependency Parsing [11], in welchem Iterationen des Malt Parsers mit angepassten Trainings- und Parsealgorithmen den ersten und fünften Platz für generelle Scores belegten. Zusätzlich zeigt der Parser solide Werte in 2010 erstellten Tests von Cer et. Al. [2].

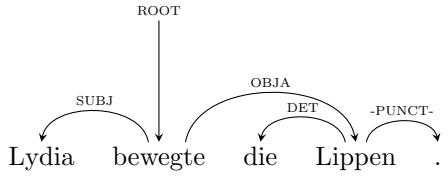

Abbildung 7: Hier wird ein Satz annotiert von Malt Parser dargestellt.

4.3 ParZu Parser

[12] Der von Mitarbeitern der Universität Zürich entwickelte ParZu Parser basiert auf der Architektur des englischen Pro3Gres Parser und bildet einen Hybrid

[11] Dies und das Folgende nach [12]
[12] Dies und das Folgende nach [16]

zwischen datengetriebenem und grammatikbasiertem Dependenz Parser. Zum Parsen eines Satzes werden sowohl eine handgeschriebene, einschränkungsbasierte Dependenz Grammatik, also auch ein, durch Maschinenlernen auf der TüBa-D/Z Treebank mit Umfang von ca. 30.000 Sätzen gewonnenes, Lexikon verwendet. Eine Liste der von ParZu verwendeten Dependenz Relations-Tags der TüBa-D/Z Treebank sind ebenfalls in Tabelle (8) und (9) im Anhang dieser Arbeit aufgeführt. Eine Ausgabe des Parsers ist in Abbildung (8) zu sehen.

Über die Leistungsfähigkeit des Parsers sind nur wenige Aussagen zu finden, allerdings zeigt ein Vergleich der Entwickler des Parsers in Sennrich et. Al. [16] ähnliche Ergebnisse zwischen ParZu und Malt Parser.

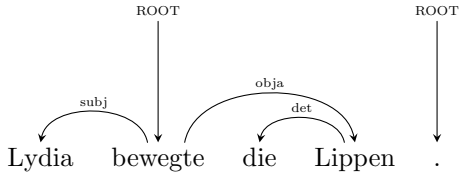

Abbildung 8: Hier wird ein Satz annotiert von ParZu Parser dargestellt.

4.4 Berkeley Parser

[13] Berkeley Parser ist ein, auf einer wahrscheinlichkeitsbasierten kontext-freien Grammatik (PCFG) basierter Konstituenz Parser, welcher starken Gebrauch von Lexika macht. Im Gegensatz zu anderen Ansätzen, werden nicht nur einzelne Eigenschaften von Wörtern betrachtet, sondern es wird ein Lexikon mit Informationen über die drei nächst wahrscheinlich folgenden Elemente für jedes Wort erstellt. Dieses Lexikon berücksichtigt jeweils das Wort selbst, sowie Part-of-Speech Tags und Nichtterminale (Konstituenzen), um den Zusammenhang zweier Wörter nach Wahrscheinlichkeiten zu bewerten. Zusätzlich besitzt die von uns verwendete Version des Berkeley Parsers, im Gegensatz zu den betrachteten Dependenz Parsern, keine Relationstags (Subjekt, Akkusativobjekt, etc.), weshalb die Auswertung des Parsers auf das Gesamtergebnis, und nicht etwa auf einzeln betrachtete Phrasen, bezogen werden muss. Diese Art der Auswertung bezeichnet man als „unlabeled attachment score“ (UAS), im Gegensatz zu LAS (labeled attachment score), nachlesbar in Kübler et. Al. [10] Kap.6.1. Die Struktur eines solchen UAS Konstituenzbaums ist in Abbildung (9) dargestellt. Zusätzlich sind die verwendeten Phrasenbezeichnungen des Parsers in Tabelle (10), sowie die Part-of-Speech Tags in Tabelle (6) und (7) im Anhang dieser Arbeit aufgeführt.

Die englische Version des Berkeley Parsers zeigt gute Leistungen in 2010 veröffentlichten Tests von Cer et. Al. [2].

[13] Dies und das Folgende nach [13]

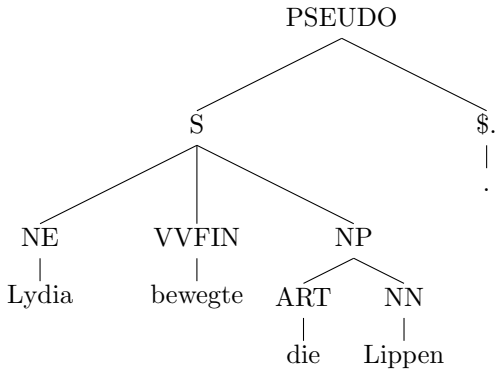

Abbildung 9: Hier wird ein Satz annotiert von Berkeley Parser dargestellt.

4.5 Stanford Parser

[14] Stanford Constituency Parser ist ein Hybrid Parser, welcher sich sowohl der Dependenz, als auch der Konstituenz Struktur bedient. Während seine Ausgabe in Form einer Konstituenz Grammatik erfolgt, benutzt der Parser allerdings mit dem Shift-Reduce Algorithmus den transitionsbasierten Ansatz eines Dependenz Parsers. Stanford Constituency Parser bearbeitet einen eingegebenen Satz wie ein Shift-Reduce Parser, mit Hilfe von „Wort Clustering“ und einem Dependenz Lexikon, ähnlich wie Mate Parser. Wenn die wahrscheinlichste Lösung für den eingegebenen Satz gefunden ist, wird diese in Konstituenz Struktur gebracht und ausgegeben. Wie bei Berkeley Parser, besitzt die im Experiment verwendete Version des Stanford Parsers keine Relationstags und erfordert deshalb eine Auswertung ohne Labels (UAS). Wie für Berkeley Parser sind die verwendeten Phrasenbezeichnungen des Parsers in Tabelle (10), sowie die Part-of-Speech Tags in Tabelle (6) und (7) im Anhang dieser Arbeit aufgelistet. Abbildung (10) enthält die Ausgabe des Parsers für einen Satz als Konstituenzbaum.

Für die Shift-Reduce Konstituenz Architektur des Stanford Parsers zeigen die Entwickler des Parsers in Zhu et. Al. [23] gute Messwerte für Englisch und Chinesisch.

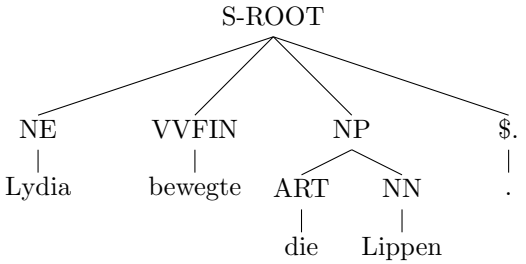

Abbildung 10: Hier wird ein Satz annotiert von Stanford Parser dargestellt.

[14] Dies und das Folgende nach [23]

5 Aufbau des Experiments

In diesem Kapitel sollen Motivation, Struktur, sowie die detailierte Vorgehensweise unseres Experiments zur Evaluation von State of the Art Syntaxparsern auf deutschen Romanen des 16. bis 20. Jahrhunderts vorgestellt werden.

5.1 Problemstellung

Traditionellerweise werden Syntax Parser auf die zu bearbeitenden Texte angepasst. Dies geschieht meist durch Verwenden einer passenden Treebank bei datengetriebenen Parsern und Lexika, oder manuelle Anpassung der Regeln von grammatikbasierten Parsern. Meist werden Parser also für eine bestimmte Art von Texten, die sog. „Domäne“, trainiert, um optimale Ergebnisse zu liefern. Dies erfordert allerdings genauere Vorkenntnisse über die Beschaffenheit der zu parsenden Texte in Sachen Satzbau, Vokabular und ähnlichem, sowie eine Spezialisierung des Parsermodells. Da das Anpassen eines Parsers auf eine Domäne also nicht immer möglich und/oder sinnvoll ist, sollen im Folgenden fünf verschiedene bereits beschriebenen State of the Art Syntax Parser auf ihre korrekten Ausgaben auf einer ihnen unbekannten Domäne getestet werden. Während die Parser auf unterschiedlichen Treebanks und/oder Grammatiken trainiert wurden, sind doch alle Parser Modelle auf per Hand annotierten Zeitungsartikeln basiert. Getestet werden sollen die Parser im Gegenzug auf deutschen Romanen des 16. bis 20. Jahrhunderts, welche sich im Allgemeinen durch eine andere Satzstruktur, sowie altertümliche, und so den Parsern unbekannte, Wörter von Zeitungsartikeln unterscheiden können.

Betrachtet wird die Treffsicherheit der Parser bezogen auf Nominalphrasen für unsere Konstituenz Parser, sowie einzelne Subjekt-, Dativobjekt-, Akkusativobjekt- und Präpositionalphrasen für unsere Dependenz Parser. Genitivobjekte werden mangels statistisch signifikanter Häufigkeitsmenge in unseren Dokumenten nicht betrachtet. Insbesondere soll eine manuelle Fehleranalyse der Ergebnisse der einzelnen Parser Aufschluss über die Problematik des Parsens auf einer unbekannten Domäne geben.

5.2 Struktur des Experiments

Das Experiment versucht einen Überblick über die Leistungsfähigkeit der einzelnen Parser in fremden Domänen zu verschaffen. Deshalb werden, falls vorhanden, die mitgelieferten Standardmodelle der Parser, trainiert auf Zeitungsartikeln, verwendet. Die zentrale Steuerung des Experiments erfolgt über Eclipse, weswegen nach Möglichkeit alle Parser über ihre Java API bedient werden.

Die zugrundeliegende Testmenge besteht aus 22 unterschiedlichen, deutschsprachigen Textausschnitten aus Romanen des 16. bis 20. Jahrhunderts mit insgesamt 10.045 Tokens. Die Länge, sowie Komplexität der Texte variiert.

Zu Begin des Experiments wurden sämtliche verwendeten Parser nach Möglichkeit an die Java API angeschlossen. Anschließend wurden für jeden Parser Methoden für die Umwandlung der Dokumente in ein passendes Eingabeformat

erstellt. Für nicht an Java anbindbare Parser wurden die Eingaben als passende .txt Dateien generiert und Methoden für die Übertragung der Ausgabewerte in die einzelnen Dokumente geschrieben.

Zusätlich zu diesen Maßnahmen musste für diejenigen Parser, welche kein vorhandenes deutsches Standardmodell besitzen, ein passendes Modell auf Basis der TüBa-D/Z Treebank der Universität Tübingen erstellt werden.

Die einzelnen Romanausschnitte, welche von den Parsern bearbeitet werden sollen, befinden sich in vorbearbeiteten .xmi Dateien, welche bereits Part-of-Speech und Morphology Tags besitzen, die von den Parsern bei Bedarf verwendet werden können. Das Auslesen, bzw. Schreiben auf diesen .xmi Dateien erfolgt über eine, von der Universität Würzburg selbst geschriebene, Java Schnittstelle in Eclipse. Geparste Sätze werden unter Kennzeichnung des Parsers über selbst geschriebene Methoden in der .xmi Datei gespeichert und können von dort jederzeit zur Analyse wieder aufgerufen werden.

Diese .xmi Dateien mussten nun zunächst mit Musterlösungen für unsere Phrasen, dem sog. „Gold Standard“, per Hand annotiert werden. Dies erfolgte über einen von Markus Krug erstellten, bisher unveröffentlichten Editor, verfügbar Online [9]. Der Editor speichert die resultierenden Annotationen direkt in den .xmi Dateien.

Als nächstes wurden für jede untersuchte Art von Phrasen Methoden erstellt, die für jedes der Parser Ergebnisse, sowie für den Gold Standard, falls möglich eine Liste mit den in der Lösung vorhandenen Phrasen ausgeben. Hier müssen also für jeden Parser Methoden zur korrekten Erstellung der Phrasen angelegt werden. Dies beinhaltet bei Dependenz Parsern die Zusammensetzung der Phrasen nach den für unseren Gold Standard ausgelegten Regeln. Für Konstituenz Parser muss die Ebene der zu vergleichenden Nominalphrasen im Konstituenzbaum festgelegt werden. Es existieren für Parser und Gold Standard jeweils Listen für Subjekte, Dativobjekte, Akkusativobjekte und Präpositionalphrasen.

Diese Listen von Phrasen ermöglichen anschließend einen direkten Vergleich von Parser und Gold Standard. Es werden für jedes einzelne Dokument die Anzahl der exakt getroffenen Phrasen als „Hits“, die Anzahl der nicht exakt getroffenen Phrasen als „Misses“, und die Anzahl der fälschlich korrekt markierten Phrasen als „False Positives“ gezählt. Durch diese Elemente kann für jeden Phrasentyp eines Dokuments eine Übersicht durch Genauigkeit(precision), Trefferquote(recall) und den aus beiden Werten berechneten F1 Score ermittelt werden, welche so ein Maß für die Treffsicherheit des Parsers geben.

Abschließend wurden die im Vergleich zum Gold Standard falsch markierten, sowie die nicht getroffenen Phrasen per Hand überprüft, um einen Eindruck über die größten Fehlerquellen der einzelnen Parser zu erhalten.

5.3 Anschließen der Parser

Bis auf ParZu Parser besitzen alle verwendeten Parser eine Java API, welche für das Anschließen genutzt wurde. ParZu selbst wurde über die Kommandozeile eines Linux Rechners bedient. Als Modell des Parsers wurde, falls vorhanden, ein mitgeliefertes Standardmodell verwendet. Bei nicht vorhandenem Standardmo-

dell wurde der Parser über die Kommandozeile auf seinen Standardeinstellungen mit Hilfe der TüBa-D/Z Treebank trainiert.

Nach korrekter Initialisierung eines Parsers, muss der zu parsende Text im korrekten Format eingegeben werden. Auch das Format des Outputs ist von Parser zu Parser verschieden, und muss gegebenenfalls für das Schreiben in die .xmi Datei umgewandelt werden.

Das Erhalten, sowie das Abspeichern der Daten wird über eine Java Schnittstelle der Universität Würzburg abgehandelt, welche Lesen und Schreiben auf den zu Grunde liegenden .xmi Dateien ermöglicht.

Stanford Constituency Parser, trainiert auf der Negra Treebank, wurde mit einem deutschen Standard Modell verwendet. Der Parser benötigt die einzelnen Sätze als Input. Part-of-Speech Tags, sowie Morphology Tags werden von Stanford selbst ausgeführt. Stanford erstellt als Konstituenz Parser sein Ergebnis als Phrasenbaum, welcher anschließend vollständig durchlaufen wird, um sämtliche Knoten einzeln mit Begin und Ende der Wörter oder Phrasen, sowie dem Phrasentyp in die .xmi Datei zu überschreiben.

Berkeley Parser besitzt, wie Stanford, ein deutsches Standard Modell. ähnlich wie Stanford, benötigt der Parser einzelne Sätze als Input und formt diese zu einem Phrasenbaum als Ausgabe. Dieser Baum wird wiederum vollständig in die .xmi Datei übertragen.

Mate Parser war bereits, wie Part-of-Speech und Morphology Tags in der .xmi Datei vorhanden. Informationen in der Datei beinhalten für jedes Wort Begin und Ende im Dokument, den zugehörigen Head der Dependenz Struktur, sowie ein Tag, welches die syntaktische Rolle des Wortes im Satz beschreibt. Das Modell des Parsers ist ebenfalls das Standardmodell, welches auf der Tiger Treebank trainiert wurde.

Malt Parser wurde mangels eines Standardmodells von uns mit einem auf der TüBa-D/Z Treebank trainieren Modell versehen und benötigte Input im CoNLL Format, was eine Satzweise Erstellung einer passenden Eingabe durch korrektes Formatieren der in den .xmi Dokumenten vorhandenen Sätze zu einer Liste aus Wörtern, Part-of-Speech, sowie Morphology Tags erforderte. Auch mussten die Morphology Tags in ein dem Parser bekanntes Format umgewandelt werden. Die resultierende Ausgabe des Parsers wird im Format eines Dependenz Graphen geliefert, welcher anschließend Wort für Wort mit Begin und Ende im Dokument, sowie dem Head des Wortes und der Dependenzrelation in die .xmi Datei übertragen wurde.

ParZu war als Linux basierter Parser nicht für die Bedienung mit unserem Windows Java System geeignet, und musste deshalb auf einem Linux System über die Kommandozeile ausgeführt und mit passendem Input versorgt werden. Für jedes zu parsende Dokument wurde ein Textdokument mit den einzelnen Wörtern und deren Part-of-Speech Tags erstellt. Als Ausgabe erstellte ParZu eine Textdatei mit den Informationen über Wortstamm, Wortart, Form, Head in der Dependenz Struktur, und syntaktischer Rolle im Satz für jedes einzelne Wort. Diese Informationen wurden anschließend per Eclipse eingelesen und in die .xmi Datei eingetragen.

5.4 Annotieren des Gold Standards

Der Testsatz des Experiments besteht aus 22 Dokumenten von verschiedenen Romanautoren des 16. bis 20. Jahrhunderts. Der Umfang einzelner Dokumente schwankt zwischen 250 und 1000 Tokens und addiert sich letztendlich zu einem Gesamtumfange von rund 10.000 Tokens. Die Anzahl der von Hand markierten Phrasen im Testsatz sind Tabelle (1) zu entnehmen.

Die zu parsenden Dokumente wurden über Markus Krug's bisher unveröffentlichten Kallimachos Annotation Editor per Hand mit folgenden Tags versehen:

Subjekt: Als Subjekt wird hier die gesamte Phrase angesehen, die im Mittelpunkt des Satzes steht. Dies bedeutet, dass alle Worte, die sich direkt auf das zentrale Subjekt beziehen, Teil des als „Subjekt" markierten Parts sind. Insbesondere müssen Aufzählungen, Genitivmodifikatoren und direkt an das Subjekt anschließende Appositionen wie Namen oder Titel berücksichtigt werden. Nicht zu einer Subjekt Phrase gezählt werden allerdings durch Kommas getrennte Appositionen.

Genitivobjekt, Dativobjekt, Akkusativobjekt: Für alle Objektphrasen verhält sich die Regelung gleich wie bei Subjekten. Genitivphrasen werden zwar für die Evaluation der Konstituenz Parser markiert, werden in dieser Arbeit allerdings nicht genauer betrachtet, da die beobachtete Menge an Dokumenten keine statistisch signifikante Anzahl solcher allein stehenden Genitivphrasen zur Verfügung stellte.

Präpositionalphrasen: Bei Präpositionalphrasen muss zunächst geprüft werden, ob es sich um alleinstehende, oder zu einem Subjekt/Objekt gehörende Phrasen handelt, da letztere zu den Subjekten/Objekten selbst hinzugezählt werden. Ferner muss, zusätzlich zu den Zusammenhangsregeln, welche bei Subjekten und Objekten gelten, geprüft werden, ob eine weitere Präpositionalphrase existiert, die sich auf die vorherige bezieht und somit Teil von ihr ist.

Nominalphrasen: Mit dem Tag „NP" werden sämtliche Nominalphrasen versehen, welche nicht in die bisher beschriebenen Kategorien der Subjekt-, Dativ-, Akkusativ-, oder Präpositionalphrasen passen. Diese werden insbesondere markiert, um den Vergleich unserer Konstituenz Parser mit unserem Gold Standard zu ermöglichen, da sowohl Stanford als auch Berkeley Parser in der von uns verwendeten Version keine Relationstags besitzen.

Phrasentyp	Subjekt	Dativobjekt	Akkusativobjekt	Präpositionalphrase
Anzahl	790	153	488	615

Tabelle 1: Hier wird die Anzahl der in der Testmenge manuell markierten Phrasen des Gold Standards aufgeführt.

5.5 Umwandlung der Parserergebnisse in Gold Standard Phrasen

Da die in unserem Gold Standard annotierten Phrasen nicht direkt aus den von unseren Parsern erstellten Bäumen herauslesbar sind, müssen die Phrasen der Parser zunächst in die für unseren Gold Standard definierten Formate gebracht werden.

Während Berkeley und Stanford Parser beide keine Relationstags besitzen und deshalb nicht über einzelne Phrasen evaluiert werden können, versuchen wir für Mate, Malt und ParZu die entstandenen Parses Phrase für Phrase mit unserem Gold Standard zu vergleichen. Da es sich allerdings bei diesen drei Parsern um Dependenz Parser handelt, müssen zuerst die Phrasen, wie in Abbildung (11) ersichtlich, aus der Satzstruktur heraus korrekt bestimmt werden.

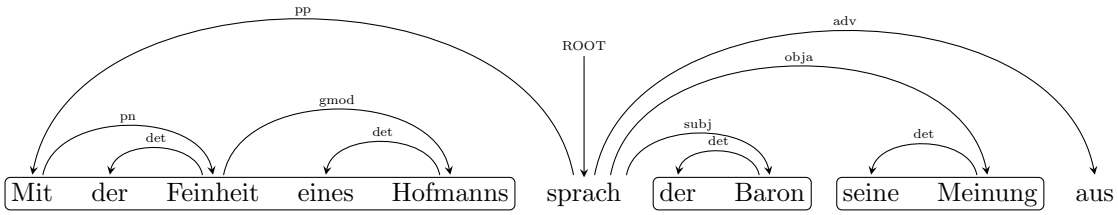

Abbildung 11: Hier wird die Unterteilung eines Dependenz Graphen in Gold Standard Phrasen dargestellt.

Wir betrachten die Parses Satz für Satz. Es müssen, nach unserer Definition der Phrasen im Gold Standard, für jeden erkannten Mittelpunkt einer Phrase - Subjekt, Dativ-, Akkusativobjekt und Präposition - die diesem Wort zugehörigen Elemente gefunden werden, um so eine Phrase zu bilden. Wir suchen also dem entsprechend für jedes Wort mit passendem Relationstag sämtliche Worte im Satz, die sich auf unser Ursprungswort beziehen, dieses also zum „Head“ haben. Anschließend müssen wir allerdings noch überprüfen, ob ein gefundenes Wort tatsächlich zu unserer Phrase dazugezählt werden kann, oder ob es sich um eine, im Gold Standard definierte, Ausnahme handelt.

Da für unterschiedliche Phrasen verschiedene Ausnahmen berücksichtigt werden müssen, benötigt jeder Phrasentyp eine eigene Methode zur Evaluation.

Der Ansatz der Evaluation allerdings ist bei allen drei Parsern gleich und kann somit übernommen werden. Es müssen lediglich die Relationstags an den entsprechenden Parser angepasst werden, da diese sich durch das Training auf verschiedenen Treebanks unterscheiden. Beispielsweise werden Subjekte von ParZu mit dem Tag „subj“, von Mate mit „SB“ und Malt mit „SUBJ“ versehen.

Zur Phrase hinzuzufügen sind bei allen Phrasen die bereits erwähnten Aufzählungen, Genitivmodifikatoren und direkt an das Subjekt anschließenden Appositionen. Zusätzlich muss hier allerdings berücksichtigt werden, dass Ketten entstehen können, deren Elemente wiederum auch zur Phrase gehören müssen.

Abbildung (12) zeigt beispielsweise eine Genitivkette, welche durch unseren Algorithmus erkannt werden muss.

Ausgenommen sind allerdings durch Kommas getrennte Appositionen, die als eigener Satzteil von ihrem Bezugswort getrennt behandelt werden sollen.

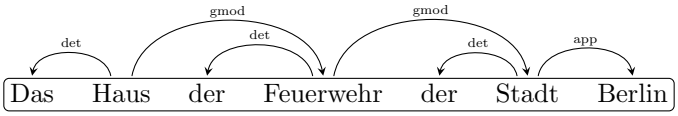

Abbildung 12: Hier wird die benoetigte Erkennung von Genitivketten als zusammengehoerige Phrase dargestellt.

Zusätzlich benötigen die einzelnen Phrasen gegebenenfalls noch spezielle Ausnahmeregeln, die sich direkt auf den Phrasentyp beziehen. So muss beispielsweise für Subjektphrasen eine Ausnahme für das „expletive es“ erstellt werden, da dieses im Tagset unserer Parser ein eigenes Tag „EXPL“ besitzt.

Die gesamte Evaluation wird für jede unserer vier Phrasen, sowie für sämtliche Dependenz Parser einzeln ausgeführt und liefert für alle Dokumente eine Liste mit Begin und Ende der einzelnen Phrasen.

5.6 Evaluation der Dependenz Parser

Die Auswertung unserer Dependenz Parser erfolgt im direkten Vergleich mit den per Hand markierten Phrasen des Gold Standards. Für jeden Parser werden die unterschiedlichen Phrasentypen einzeln ausgewertet, indem über alle Dokumente iteriert wird. Um also beispielsweise die Leistung des Malt Parsers beim markieren von Subjekten in einem Dokument zu untersuchen, werden zunächst zwei Listen erstellt, welche Begin und Ende der entsprechenden Phrasen nach Gold Standard und Malt Parser enthalten. Nun werden beide Listen abgeglichen. Ein Treffer bezeichnet hierbei eine Phrase, für die Begin und Ende in beiden Listen übereinstimmen. Aus der Länge der einzelnen Listen, zusammen mit der Anzahl der Treffer, lassen sich nun als Qualitätsmaße der Parser Precision (Genauigkeit) und Recall (Trefferquote) für jedes einzelne Dokument ermitteln, die wie folgt berechnet werden:

Precision (Genauigkeit) bezeichnet die Treffsicherheit des Parsers und wird berechnet durch das Teilen der tatsächlich korrekten Treffer mit den vom Parser Vermuteten. Es beschreibt also wie oft ein Parser mit seiner Lösung richtig liegt.

Recall (Trefferquote) bezeichnet die Rate an korrekt gefundenen Phrasen und wird durch das Teilen der korrekten Treffer durch die in der Musterlösung vorhandenen Phrasen berechnet. Es beschreibt also wie viele korrekte Phrasen ein Parser gefunden hat.

Beide Maße können durch bilden des harmonischen Mittelwerts miteinander verrechnet werden um so ein Maß für die Leistung des Parsers auf beiden Gebieten zu erhalten. Diese Verrechnung nennt sich „F1 score“.

Diese Maße werden für alle 22 vorhandenen Dokumente erstellt. Für unsere Ergebnisse werden sie schließlich, gewichtet mit der Anzahl der vorkommenden Phrasen im Gold Standard, verrechnet, um einen gewichteten Durchschnitt (Mikro Average) der Treffsicherheit des Parsers für den entsprechenden Phrasentyp zu erhalten.

5.7 Evaluation der Konstituenz Parser

Da beide unserer verwendeten Konstituenz Parser keine Tags besitzen, welche die Funktion der Phrasen im Satz beschreiben, müssen wir für die Evaluation dieser Parser auf andere Vergleichsmöglichkeiten zurückgreifen.

Es werden alle verfügbaren Phrasen des Gold Standards zusammengelegt und als Nominalphrasen (NP) betrachtet. Zusätzlich werden hierfür noch in der Musterlösung alle bisher nicht abgedeckten Nominalphrasen, wie zum Beispiel Konjunktionalgruppen, Zeitangaben, etc. mit dem Tag „NP“ versehen, um so die Lösungen der Parser auf Basis ihrer Nominalphrasen vergleichen zu können.

Um nun aus den Lösungsbäumen der Parser die passenden Nominalphrasen zur Überprüfung auszulesen, gehen wir wie folgt vor:

Für jedes Wort innerhalb eines Satzes wird die kleinstmögliche Phrase betrachtet, welche dieses Wort enthält und ein Tag besitzt, das nicht „S“, „CS“, „PSEUDO“, „“, oder „VP“ lautet. Für den Fall, dass sich Phrasen überlappen, wird nur die längste Phrase behalten. Es wird also die jeweils kleinste gültige Unterteilung aller Nominalphrasen als Lösung des Parsers angesehen. Dies ist die am häufigsten zutreffende Annahme, welche wir über die Lösungsbäume der geparsten Dokumente treffen konnten.

Für jedes Dokument wird anschließend eine Liste mit den so entstehenden Phrasen angelegt, welche somit sämtliche gefundenen Nominalphrasen des Parsers enthält. Aus unserem Gold Standard wird eine ähnliche Liste für jedes Dokument erstellt, die lediglich Begin und Ende jeder markierten Nominalphrase enthält. Dies ermöglicht im Anschluss einen direkten Vergleich der beiden Listen auf korrekt markierte Phrasen, nicht erkannte Phrasen, sowie fälschlich markierte Phrasen und die resultierende Erstellung von Precision, Recall und F1 Score.

6 Ergebnisse des Experiments

Im Folgenden soll auf die Ergebnisse unseres Experimentes eingegangen werden. Berücksichtigt werden sollen die Durchschnittstrefferquoten der Parser, die überzeugendsten Ergebnisse für die verschiedenen Phrasentypen, sowie die Trefferraten auf unterschiedlichen Dokumenten.

Tabelle (2) zeigt den Gesamtdurchschnitt der Ergebnisse von Precision, Recall und F1 Score für unsere beiden Konstituenz Parser, deren sog. „Makro Average“, sowie den anhand der einzelnen Phrasen gewichteten Gesamtdurchschnitt der Dependenz Parser, den sog. „Mikro Average“.

Parser Average	**Berkeley**	**Stanford**	**Mate**	**Malt**	**ParZu**
Precision p	56,00%	59,00%	65,72%	67,22%	74,40%
Recall r	46,00%	48,00%	62,71%	61,78%	70,77%
F1 score	**51,00%**	**51,00%**	**64,07%**	**64,29%**	**72,36%**

Tabelle 2: Hier wird der sog. „Weighted Average“ aller Parser auf unserer Testmenge aufgelistet.

Auf den ersten Blick kann zunächst festgestellt werden, dass ParZu einen enormen Vorsprung zu sämtlichen anderen Parsern besitzt. Dies bedeutet, dass ParZu für die betrachteten Dokumente eindeutig bessere Ergebnisse liefert, sofern man genau nach Subjekten, Dativ-, Akkusativobjekten und Präpositionalphrasen unterteilen will. Für weitere Rückschlüsse müssen allerdings die einzelnen Parser genauer betrachtet werden, da im Falle unserer Abbildung beispielsweise ein schlechter Wert bei der Erkennung von Akkusativobjekten das Ergebnis stark verzerrt.

Auch fällt auf, dass beide unserer Konstituenz Parser weit hinter den Dependenz Parsern zurückliegen. Dies kann allerdings keinesfalls ausschließlich auf die Architektur der Parser zurückgeführt werden, sondern ist mit höherer Wahrscheinlichkeit der Tatsache zuzuschreiben, dass sowohl Berkeley, als auch Stanford Parser in ihrer hier verwendeten Form keine Syntax Relationen vermerken, und so die Auswertung anders erfolgen musste.

Um diese Ergebnisse nun weiter auszuwerten und allgemeinere Schlüsse zu ziehen, wollen wir die Resultate auf die einzelnen Parser bezogen, und nach Möglichkeit auf die einzelnen Phrasen aufgeteilt betrachten. Auch sollen die Ergebnisse einiger interessanter Dokumente betrachtet werden, bei denen beispielsweise deutliche Schwankungen der Parsereffizienz beobachtet werden können.

Dependenz Mikro Average		**Subjekt**	**Dativobjekt**	**Akkusativobjekt**	**Präpositional phrase**
Mate	p	72,64%	44,92%	55,31%	70,28%
	r	67,22%	34,64%	60,86%	65,37%
	F1	**69,82%**	**39,11%**	**57,95%**	**67,73%**
Malt	p	75,62%	55,14%	58,28%	66,55%
	r	69,87%	38,56%	58,40%	59,84%
	F1	**72,63%**	**45,38%**	**58,34%**	**63,01%**
ParZu	p	80,45%	72,39%	71,50%	69,45%
	r	72,93%	87,58%	61,68%	71,71%
	F1	**76,22%**	**79,26%**	**66,23%**	**70,56%**

Tabelle 3: Hier werden die Ergebnisse der Dependenz Parser auf die verschiedenen betrachteten Phrasen aufgeteilt. Die Tabelle zeigt die sog. „Mikro Averages“ des Experiments.

6.1 Ergebnisse der einzelnen Parser

Da wir die Informationen über die Treffsicherheit der Dependenz Parser bei einzelnen Phrasentypen, dargestellt in Tabelle (3), besitzen, können wir diese getrennt betrachten und so einen Rückschluss auf deren gesamten Weighted Average aus Tabelle (2) ziehen.

Zunächst stellen wir fest, dass die Ergebniswerte der Subjekte durchgehend höher als andere Phrasen sind. Auch sehen wir hier, insbesondere bei der Betrachtung der Genauigkeit (Precision p), einen deutlichen Leistungsunterschied der Parser.

Besonders interessant sind die Ergebnisse der Dativobjekte, die von Parser zu Parser stark variieren und bei Mate und Malt sehr schwach, bei ParZu jedoch vergleichsweise hervorragend ausfallen. Dies kann zunächst einmal daran liegen, dass die Menge der Dativobjekte in unseren Testdokumenten einigermaßen gering ausfällt (150 Dativobjekte im Gegensatz zu 790 Subjekten), und diese Phrasen hauptsächlich in längeren, und dadurch für die Parser komplizierteren Sätzen auftreten. Die Ergebnisse des ParZu Parsers hier sind beeindruckend, stemmen allerdings hauptsächlich von einigen wenigen Dokumenten, welche viele Dativobjekte der selben Form enthalten.

Weiterhin erstaunlich ist das schwache Resultat aller Parser für Akkusativobjekte, speziell im Gegensatz zu den deutlich höheren Werten der Subjekte. Selbst der andernfalls hervorragende ParZu Parser trifft hier, anhand der Trefferquote (Recall r) zu erkennen, einen recht großen Teil der markierten Phrasen nicht.

Für Präpositionalphrasen scheinen sämtliche Parser wieder deutlich effizienter zu arbeiten und liefern teils mit den Resultaten für Subjekte vergleichbare Ergebnisse.

Generell kann durch diese Ergebnisse vermerkt werden, dass ParZu auf unserer Testmenge in sämtlichen Kategorien mit den besten Durchschnittswerten (F1 score) aufwartet. Interessant ist zudem, dass unterschiedliche Parser Probleme an verschiedenen Stellen zeigen. So stellt beispielsweise für ParZu die Erkennung von Akkusativobjekten die größte Herausforderung dar, wärend Malt und Mate ihre Tiefstwerte bei Dativobjekten besitzen.

6.2 Ergebnisse einzelner erwähnenswerter Dokumente

In beiden bereits analysierten Tabellen konnten große Schwankungen und Unterschiede zwischen den einzelnen Parsern festgestellt werden. Um den Ursprung dieser Unregelmäßigkeiten zu finden, möchten wir nun auf einige der erwähnten Ursachen eingehen, indem wir unter diesen Aspekten Precision, Recall und F1 Score einzelner Dokumente betrachten.

Für unsere Konstituenz Parser ist zunächst zu beobachten, dass sich für sämtliche Dokumente alle Werte ähneln. Es können durchschnittlich Unterschiede von 5% im F1 Score festgestellt werden, was ähnliche Schwierigkeiten beim Parsen für Stanford und Berkeley vermuten lässt. Auch schwanken die Gesamtwerte von einem Maximum von rund 70% bei syntaktisch simplen Dokumenten

bis hinunter zu rund 38% für Dokumente mit langen, kompliziert verschachtelten Sätzen. Dieses Phänomen kann bei beiden Parsern beobachtet werden. Generell bleiben die Ergebnisse allerdings in relativer Nähe zum Gesamtdurchschnitt aus Tabelle (2).

F1 Score	**Gotthelf,-Jeremias Uli der Pächter**				**Knigge,-Adolph-Freiherr-von Josephs von Wurmbrands ... politisches Glaubensbekenntnis**			
	SB	**DA**	**OA**	**PP**	**SB**	**DA**	**OA**	**PP**
Mate	85%	60%	50%	73%	70%	30%	48%	52%
Malt	85%	67%	57%	50%	75%	22%	50%	53%
ParZu	86%	96%	78%	71%	70%	67%	57%	69%

Tabelle 4: Hier wird der F1 Score der Dependenz Parser für zwei ausgewählte Dokumente der Testmenge dargestellt.

Für die genauere Betrachtung der Dependenz Parser fokusieren wir uns an dieser Stelle auf einige Dokumente, welche in ihren Ergebnissen der einzelnen Phrasen große Schwankungen zwischen den Parsern enthalten. Generell kann festgestellt werden, dass die Treffsicherheit der Parser mit der Länge und Komplexität der zu parsenden Sätze deutlich abnimmt. Ein Beispiel dieses Verhaltens weisen die in Tabelle (4) abgebildeten Dokumente auf. Während der Textausschnitt von „Uli der Pächter“ eine simple Satzstruktur in direkter Rede aufweist, ist der zweite Text mit Nebensätzen und Semicola deutlich komplizierter formuliert. Wie dies, in Verbindung mit der Nutzung von den Parser Lexika unbekannten Worten, auf das Ergebnis zurückzuführen ist, soll im folgenden Kapitel der genaueren Fehleranalyse an den erwähnten Dokumente erläutert werden.

Für die Untersuchung der Konstituenz Parser werden ebenfalls beide bereits verwendeten Dokumente betrachtet. Während Stanford Parser, in Tabelle (5) ersichtlich, vergleichsweise gute Werte für Dokument 1 vorweisen kann, kann eine enorm schwache Trefferquote (Recall) von nur 28% für Dokument 2 beobachtet werden. Auch Berkeley Parser gibt mit 40% eine sehr schwache Trefferquote

		Gotthelf,-Jeremias Uli der Pächter	**Knigge,-Adolph-Freiherr-von Josephs von Wurmbrands ... politisches Glaubensbekenntnis**
Berkeley	p	69%	64%
	r	58%	40%
	F1	**63%**	**49%**
Stanford	p	81%	60%
	r	70%	28%
	F1	**74%**	**38%**

Tabelle 5: Hier werden Precision p, Recall r, und F1 Score F1 der Konstituenz Parser für zwei ausgewählte Dokumente der Testmenge dargestellt.

an. Dies weißt auf enorme Schwierigkeiten für beide Parser bei kompliziertem, langem Satzbau hin und soll im Folgenden ebenfalls genauer analysiert werden.

6.3 Fehleranalyse

Wir möchten nun die in Tabelle (4) und (5) dargestellten Ergebnisse genauer betrachten und gegebene Schwankungen auf ihren Ursprung zurückführen. Dies soll insbesondere dafür genutzt werden, um zu entscheiden, wie sinnvoll der Einsatz eines bestimmten Parsers für ein gegebenes Dokument ist.

6.3.1 Fehleranalyse: Dependenz Parser

Zuerst wollen wir die Dependenz Parser genauer untersuchen, welche durch ihre Relationstags einen genaueren Einblick in die Treffsicherheit bei einzelnen Phrasentypen ermöglichen.

Bei genauerer Untersuchung der Subjekte wird ersichtlich, dass die Lösungen unserer Parser von Grund auf verschieden sind. Zwar sind Fehler auf den selben Phrasen erkennbar, doch scheinen die gelieferten Lösungsbäume sich bei komplizierteren Phrasen oftmals deutlich zu unterscheiden.

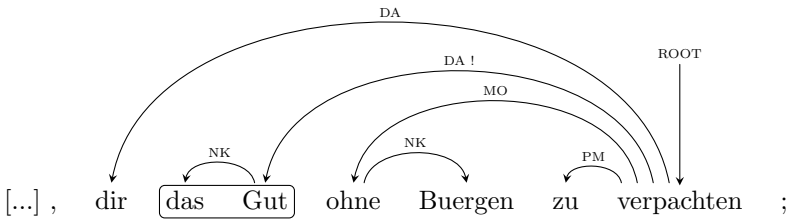

Abbildung 13: Diese Abbildung zeigt einen Fehler des Parsers im Markieren inkorrekter Relationen. (Markiert mit !)

Ein häufig auftretender Fehler bei Mate und Malt Parser ist das Setzen des falschen Dependenz Relationen. So werden oftmals Subjekte als Akkusativobjekte, aber auch Dativobjekte als Subjekte bezeichnet. Dieser Fehler tritt zwar bei allen drei Parsern ein, bei ParZu allerdings in geringerem Ausmaß. Die in Abbildung (13) dargestellte Markierung eines Akkusativobjekts als Dativobjekt beispielsweise, scheint bei ParZu nicht gehäuft vorzukommen. Dies ist ein besonders herausragender Fehler, da viele Dativobjekte anhand ihrer Endung eindeutig zu identifizieren sind. Im Falle der Abbildung (13) bedeutet dies, dass Mate und Malt Parser für die Phrase „das Gut“ die Entscheidung über die Art der Phrase ohne genaue Berücksichtigung der morphologischen Daten beider Wörter treffen. ParZu allerdings scheint sich stärker an die morphologischen Informationen anzulehnen und markiert die Phrase korrekt.

Ebenfalls eine häufige Fehlerquelle, speziell für unsere Domäne, stellt das Vorkommen unbekannter Wörter dar. Die in unseren Dokumenten zwar nur

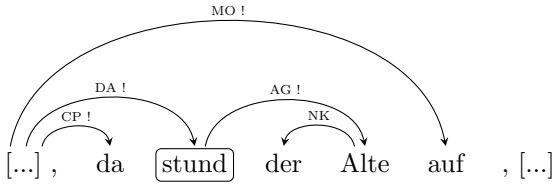

Abbildung 14: Diese Abbildung zeigt Folgefehler des Parsers durch Nichterkennen des Wortes im Satzmittelpunkt. (Falsche Relationen markiert mit !)

eingeschränkt vorkommenden, aber dennoch vorhandenen altertümlichen oder umgangssprachlichen Wörter befinden sich nicht in den Lexika der Parser, und erfordern somit eine Bearbeitung ohne nützliche Zusatzinformationen. Das Beispiel in Abbildung (14) zeigt das den Parsern nicht bekannte Wort „stund“, welches allerdings einen zentralen Bestandteil des Satzes darstellt. Während Mate das Wort „stund“ selbst, trotz Kleinschreibweise, als Dativobjekt markiert, setzen sowohl Malt als auch ParZu das „ROOT“ Tag für Prädikate im Satzmittelpunkt zwar korrekt, weisen „der Alte“ allerdings Demselben als Genitivmodifikator zu. Dies lässt vermuten, dass das „ROOT“ Tag aufgrund des Nichterkennen des Wortes, und nicht wegen korrekter Identifikation desselben als Satzmittelpunkt gesetzt wurde. Die entstehenden Probleme sind ebenfalls in Abbildung (14) ersichtlich. Der gesamte, auf dem Prädikat „stund“ basierende Nebensatz wird in Abhängigkeit eines anderen, nicht zum Satz gehörenden Prädikats gestellt und enthält deshalb viele fälschlich markierte, hier mit „!“ verdeutlichte Tags. Darüber hinaus zeigt die Abbildung deutlich, dass die falsche Etikettierung eines Wortes zu Folgefehlern führen, und sich so negativ auf das Gesamtergebnis auswirken kann.

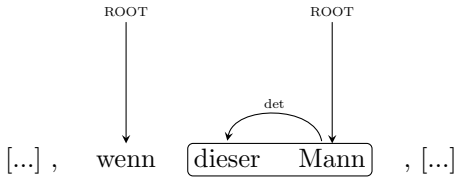

Abbildung 15: Diese Abbildung zeigt einen Fehler des Parsers im Setzen der Relation „ROOT“ bei nicht kategorisierbarem Wort.

Ein häufig gesehener Fehler bei ParZu und Malt Parser ist das Setzen des „ROOT“ Tags Anstelle von erwarteten Tags für Subjekte oder Objekte. Dies kann mehrere Gründe haben. Zum einen kann ein „ROOT“ Tag gesetzt werden, falls das Wort dem Lexikon des Parsers gänzlich unbekannt ist und dieses ebenfalls nicht in die Struktur des Satzes zu bringen ist. Zum anderen sind „ROOT“ Tags auch möglich, falls das Wort aus anderen Gründen nicht in die Struktur des Satzes eingegliedert werden kann, beispielsweise da die Struktur des Sat-

zes zu kompliziert ist. In Abbildung (15), werden sowohl das Wort „wenn“, als auch das Wort „Mann“ von ParZu als „ROOT“ markiert. Der mutmaßliche Grund für diesen Fehler ist hier die Position des Textausschnittes in Mitten von mehreren Nebensätzen, welche ParZu die syntaktische Analyse erschweren. Der Ausschnitt steht an Tokenstelle 220 des Satzes, während der gesamte Satz eine Länge von knapp 300 Tokens besitzt. Generell scheint ParZu sehr schnell unsichere Elemente mit „ROOT“ Tags zu versehen, während im Gegensatz Mate für sämtliche Elemente ein passendes Tag zu finden versucht. Dies kann für Mate zum Erfolg führen, senkt allerdings bei falscher Einordnung zusätzlich die Genauigkeit des Parsers für die entsprechend fälschlich gesetzte Phrase. Dies kann auch in den Ergebnissen der Parser beobachtet werden, bei denen ParZu deutlich höhere Durchschnittswerte für Genauigkeit zeigt, als andere Parser.

Bei näherer Betrachtung der Dativobjekte wird die bereits besprochene fehlende Berücksichtigung der Morphology Tags bei Mate und Malt noch deutlicher. Hier ist bei beiden Parsern eine massive Senkung der Trefferquote zu erkennen, was hauptsächlich auf die bereits in Abbildung (13) aufgeführte Problematik zurückzuführen ist. Es werden Dative fälschlich als Subjekte und Akkusative markiert. Auch sind einige fälschlicherweiße als Dative markierte Subjekte und Akkusative auffindbar. Während Malt durch das setzen von „ROOT“ Tags bei einigen Phrasen noch eine bessere Genauigkeit aufweisen kann, scheint Mate enorme Schwierigkeiten mit Dativen zu besitzen. ParZu zeigt im Gegensatz zu den anderen Parsern, keine großartigen Probleme mit der Verwechslung von Dativen, Subjekten und Akkusativen, und behält die vergleichsweise hervorragenden Werte der Subjektphrasen bei. Auch ist, gerade im Vergleich der beiden in Tabelle 4 abgebildeten Dokumente, ein sehr starker Abfall der Treffsicherheit der Parser bei komplizierterem Satzbau zu erkennen. Viele Fehler entstehen hier nicht nur aus der Morphology Problematik, sondern auch aus einer falschen Zuordnung der Phrasen zu nicht vorhandenen Aufzählungen, Appositionen, oder ähnlichen Konstruktionen. Hier zeigen die Parser deutliche Schwierigkeiten mit verschachtelten Nebensätzen.

Akkusativobjekte weisen ähnliche Probleme wie unsere Dativobjekte auf. Die Problematik des Vertauschens von Tags besteht weiterhin, was ParZu durch die Tendenz zum Setzen der „ROOT“ Tags deutlich bessere Treffsicherheit einbringt. Für unser zweites Dokument allerdings zeigt selbst ParZu starke Schwierigkeiten mit der Zuweisung der Phrasen, was auch hier wieder auf den komplizierten Satzbau zurückgeführt werden kann.

Für Präpsitionalphrasen besitzt Dokument 1 insgesamt nur 5 Phrasen, was das Ergebnis somit relativiert. Erkennbar sind hier allerdings leichte Schwierigkeiten der Parser im Umgang mit altertümlicher direkter Rede, da hier unbekannte Worte in Kombination mit Semicola und anderweitig ungewöhnlicher Satzstruktur zu fälschlich markierten Funden, sog. „false positives“ führen. Für Dokument 2 sind, bei größerer Testmenge von 32 zu findenden Phrasen, wiederum aufs Neue die Probleme der anderen Objekte bei Vertauschen von Tags, sowie die generelle Problematik der komplizierten Satzstruktur zu erkennen.

Abschließend kann gesagt werden, dass sämtliche Dependenz Parser für unsere Domäne verwendbare Ergebnisse liefern. Die Resultate, allerdings, zeigen

deutlich schlechtere Ergebnisse im Vergleich zu bereits erwähnten Messungen der einzelnen Parser auf korrekt angepassten Domänen. Für das Parsen von Dativobjekten scheint allein ParZu verlässliche Ergebnisse zu liefern. Auch insgesamt scheint ParZu, insbesondere durch die korrekte Verwendung von Morphology Tags, sowie das häufige Setzen von „ROOT“ Tags die verlässlichsten Ergebnisse des Experiments zu liefern. Darüber hinaus besonders zu beachten sind bei Wahl eines passenden Parsers, oder auch bei gezieltem Trainieren eines Parsers auf eine Domäne, die Komplexität, sowie generelle Länge der Satzstruktur.

6.3.2 Fehleranalyse: Konstituenz Parser

Die in unserem Experiment verwendeten Konstituenz Parser besitzen keine Relationstags und können somit nicht direkt zwischen einzelnen Nominalphrasentypen unterscheiden. Die Evaluation erfolgt daher nur allgemein unter Berücksichtigung aller Nominalphrasen.

Als besonders gravierend wurde bereits bei der Analyse einzelner Dokumente der starke Einbruch der Trefferquote beider Parser bei komplizierterem Satzbau betrachtet. Dieses Phänomen stellt bei genauerer Betrachtung ein großes Problem für die Anwendung beider Parser auf unsere zu testende Domäne dar. Sowohl Stanford, als auch Berkeley Parser scheinen bei einer zu komplizierten Satzstruktur fehlerhaft große Mengen an Tokens in eine Nominalphrase zusammenzufassen und somit sämtliche innerhalb dieser Menge enthaltene Nominalphrasen zu ignorieren. So wurde beispielsweise der komplette erste Satz des zweiten genauer betrachteten Dokuments als eine Nominalphrase bezeichnet. Mutmaßlicher Grund für diese falsche Annahme ist hier die enorme Länge des Satzes von insgesamt 295 Tokens. Dies hat allerdings für die Trefferquote der Parser gravierende Folgen, da sämtliche Nominalphrasen innerhalb des Satzes nicht als Lösungen geliefert wurden, was für den Beispielsatz allein einen Verlust von über 30 Nominalphrasen bedeutet, welche dem zur Folge als nicht korrekt markierte Phrasen (Misses) eingetragen werden müssen. Auch scheint sich die fälschliche Zuordnung von Nominalphrasen nicht auf gesamte Sätze beschränken. So beinhaltet beispielsweise der zweite Satz des Dokuments eine von Berkeley Parser erstellte Nominalphrase an Stelle eines Nebensatzes. Während sich dieser Fehler bei vielen der Dokumente nur selten zeigt, ist er dennoch für einen enormen Verlust an Trefferquote durch die schlechten Ergebnisse bei komplizierter formulierten Dokumenten verantwortlich.

Darüber hinaus zeigen beide Parser leichte Schwierigkeiten mit ihnen wohl unbekannten Satzzeichen. Gerade die direkte Rede des ersten Dokuments scheint ihnen Probleme zu bereiten. Auch hier können wieder fälschlich gesetzte Verknüpfungen zwischen Sonderzeichen zu einem Verlust der zwischen den Zeichen befindlichen Nominalphrasen führen.

Des weiteren kann noch bemerkt werden, dass Stanford Parser Schwierigkeiten besitzt, von einander unabhängige, aber dennoch aufeinander folgende Phrasen zu erkennen. Genauer bezieht sich das Problem auf Präpositionalphrasen, die sich auf das Verb des Satzes, nicht etwa auf ein Nomen beziehen. So

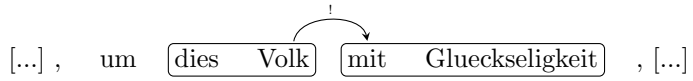

Abbildung 16: Diese Abbildung zeigt einen Fehler des Parsers im markieren des falschen Bezugsworts von Praepositonalphrasen.

wird beispielsweise die Präpositionalphrase „mit Glückseligkeit“ nicht etwa, wie in Abbildung (16) durch die Umrandungen korrekt dargestellt als unabhängige Phrase auf das Verb des Satzes, sondern fälschlicherweise wie durch den Relationspfeil auf das Akkusativobjekt „dies Volk“ bezogen.

Abschließend kann beobachtet werden, dass unsere Konstituenz Parser ohne Relationstags enorme Schwierigkeiten mit komplizierter Satzstruktur und generell lang ausgeführten Sätzen zeigen. Sie zeigen somit für die Analyse der hier gewählten Domäne, durch die Durchschnittswerte des Experiments bestätigt, nur begrenzte Eignung.

7 Zusammenfassung

Während sämtliche in unserem Experiment verwendeten Parser gute Werte in externen Tests auf einer passenden Domäne vorweisen, ist im Bezug auf die, den Parsern unbekannte, Domäne der Romane des 16. bis 20. Jahrhunderts deutlich schwächere Treffsicherheit bei allen Parsern zu erkennen. Als besonders treffsicher stellte sich der ParZu Parser der Universität Zürich heraus, welcher sich mit veringerter Fehlerhäufigkeit durch das korrekte Nutzen von Morphology Tags, sowie das bereitwillige Setzen von „ROOT“ Tags bei unbekannten, oder generell nicht zuordenbaren Worten von den anderen Parsern absetzte. Große Schwierigkeiten hingegen zeigten die verwendeten Konstituenz Parser, welche wegen mangelnder Relationstags einzig basierend auf ihre Nominalphrasen ausgewertet wurden, und hierbei das fälschliche Markieren einzelner Sätze als Nominalphrasen zum Verlust einer großen Menge an erkennbaren Phrasen führte. Allgemein ist zu beobachten, dass sämtliche Parser deutliche Schwierigkeiten mit langen und kompliziert verschachtelten Sätzen zeigen, welche in unserer Testmenge an Romanen eher zahlreich, in den auf Zeitungsausschnitten basierten Trainingsmodellen der Parser allerdings weniger häufig auftreten. Dieser Aspekt, sowie die Vorkommenshäufigkeit von altertümlichen und umgangssprachlichen, und so den Parsern meist unbekannten, Worten machen einen Großteil der beobachteten Fehler der Testmenge aus und sollten somit bei der Wahl des Parsers für eine bestimmte Domäne berücksichtigt werden. Eine Verbesserung der Leistung unserer Parser könnte beispielsweise durch die im CoNLL 2007 shared task diskutierten Ansätze zur automatischen Domänenanpassung erreicht werden. Auch scheint die Verwendung von Konstituenz Parsern mit vorhandenen Relationstags sinnvoll, um so zu versuchen die Fehlerquelle des Markierens ganzer Sätze als Nominalphrasen zu umgehen.

8 Anhang

STTS Tags 1	Beschreibung	Erlaeuterung
ADJA	attributives Adjektiv	[das] große [Haus]
ADJD	adverbiales oder praedikatives Adjektiv	[er faehrt] schnell [er ist] schnell
ADV	Adverb	schon, bald, doch
APPR	Praeposition; Zirkumposition links	in [der Stadt], ohne [mich]
APPRART	Praeposition mit Artikel	im [Haus], zur [Sache]
APPO	Postposition	[ihm] zufolge, [der Sache] wegen
APZR	Zirkumposition rechts	[von jetzt] an
ART	bestimmter oder unbestimmter Artikel	der, die, das, ein, eine, ...
CARD	Kardinalzahl	zwei [Maenner], [im Jahre] 1994
FM	Fremdsprachliches Material	[Er hat das mit „“] A big fish [„“ übersetzt]
ITJ	Interjektion	mhm, ach, tja
ORD	Ordinalzahl	[der] neunte [August]
KOUI	unterordnende Konjunktion mit „zu“ und Infinitiv	um [zu leben], anstatt [zu fragen]
KOUS	unterordnende Konjunktion mit Satz	weil, daür, damit, wenn, ob
KON	nebenordnende Konjunktion	und, oder, aber
KOKOM	Vergleichskonjunktion	als, wie
NN	normales Nomen	Tisch, Herr, [das] Reisen
NE	Eigennamen	Hans, Hamburg, HSV
PDS	substituierendes Demonstrativpronomen	dieser, jener
PDAT	attribuierendes Demonstrativpronomen	jener [Mensch]
PIS	substituierendes Indefinitpronomen	keiner, viele, man, niemand
PIAT	attribuierendes Indefinitpronomen ohne Determiner	kein [Mensch], irgendein [Glas]
PIDAT	attribuierendes Indefinitpronomen mit Determiner	[ein] wenig [Wasser], [die] beiden [Brueder]
PPER	irreflexives Personalpronomen	ich, er, ihm, mich, dir
PPOSS	substituierendes Possessivpronomen	meins, deiner
PPOSAT	attribuierendes Possessivpronomen	mein [Buch], deine [Mutter]
PRELS	substituierendes Relativpronomen	[der Hund ,] der
PRELAT	attribuierendes Relativpronomen	[der Mann ,] dessen [Hund]
PRF	reflexives Personalpronomen	sich, einander, dich, mir

Tabelle 6: Obige Grafik enthält Teil 1 der Part-of-Speech Tagset Tabelle des STTS der Universitäten Stuttgart und Tübingen, entnommen Online [21].

STTS Tags 2	Beschreibung	Erlaeuterung
PWS	substituierendes Interrogativpronomen	wer, was
PWAT	attribuierendes Interrogativpronomen	welche [Farbe], wessen [Hut]
PWAV	adverbiales Interrogativ- oder	
Relativpronomen	warum, wo, wann, worueber, wobei	
PAV	Pronominaladverb	dafür, dabei, deswegen, trotzdem
PTKZU	„zu“ vor Infinitiv	zu [gehen]
PTKNEG	Negationspartikel	nicht
PTKVZ	abgetrennter Verbzusatz	[er kommt] an, [er faehrt] rad
PTKANT	Antwortpartikel	ja, nein, danke, bitte
PTKA	Partikel bei Adjektiv oder Adverb	am [schoensten], zu [schnell]
SGML	SGML Markup	
SPELL	Buchstabierfolge	S-C-H-W-E-I-K-L
TRUNC	Kompositions-Erstglied	An- [und Abreise]
VVFIN	finites Verb, voll	[du] gehst, [wir] kommen [an]
VVIMP	Imperativ, voll	komm [!]
VVINF	Infinitiv, voll	gehen, ankommen
VVIZU	Infinitiv mit „zu“, voll	anzukommen, loszulassen
VVPP	Partizip Perfekt, voll	gegangen, angekommen
VAFIN	finites Verb, aux	[du] bist, [wir] werden
VAIMP	Imperativ, aux	sei [ruhig !]
VAINF	Infinitiv, aux	werden, sein
VAPP	Partizip Perfekt, aux	gewesen
VMFIN	finites Verb, modal	duerfen
VMINF	Infinitiv, modal	wollen
VMPP	Partizip Perfekt, modal	gekonnt, [er hat gehen] koennen
XY	Nichtwort, Sonderzeichen enthaltend	3:7, H2O, D2XW3
$,	Komma	,
$.	Satzbeendende Interpunktion	. ? ! ; :
$(	sonstige Satzzeichen; satzintern	- [,] ()

Tabelle 7: Obige Grafik enthält Teil 2 der Part-of-Speech Tagset Tabelle des STTS der Universitäten Stuttgart und Tübingen, entnommen Online [21].

Dependenz Relation Bezeichnung	**TueBa -D/Z (Malt, ParZu)**	**Tiger (Mate)**	**Beschreibung/Beispiel**
Adjektivkomponente		ADC	
Adverbiale Modifikation	ADV		
Massangabe eines Adjektivs		AMS	*Eine [Woche] spaeter*
Apposition	APP	APP	verbindet aufeinanderfolgende Worte derselben NP, falls sie nicht Determiner oder Attribute sind
Attribut	ATTR		Attribut eines Nomens
Verbgruppe	AUX		Verbgruppe aus Hilfsverb und Vollverb
Adverbialphrasen-Komponente		AVC	
Getrenntes Verb Praefix	AVZ	SVP	*auf, an, ab, aus, ein*
Komparatives Komplement		CC	Teilelement eines Satzes mit komparativer Konjunktion
nebengeordnete Konjunktion		CD	Konjuntionen: und, oder ; Teilelemente der Konjunktion werden als NK und MO markiert
Komplement einer Konjunktion	CJ	CJ	Letztes Wort einer Beiordnung
komparative Konjunktionen		CM	Konjunktionen: als, wie ; Teilelemente der Konjunktion werden als NK und MO markiert
Komplementierer		CP	*ich will wissen, [ob] etwas passiert ist*
Funktionsverbgefuege		CVC	*[in] Kraft treten*
Determiner eines Nomens	DET		Artikel und attributive Pronomen
Begin direkter Rede		DH	
Anfuehrungszeichen		DM	
Freier Dativ	ETH		Entspricht nicht dem normalen Verbrahmen
Expletives „es“	EXPL		
Genitivattribut	GMOD	AG	*Der Herr [des Hauses]*
Nominalphrase als Maßangabe	GRAD		*Der drei [Jahre] alte Clio*
Junktor		JU	
Vergleichsworte „wie“ und „als“	KOM		
Teil einer Beiordnung	KON		Ende der Beiordnung wird as CJ bezeichnet, alle anderen Teile als KON
untergeordnete Konjunktion	KONJ		Verbindet Konjunktion mit Bezugswort
Postnominaler Modifikator		MNR	*im Traume, [ohne] Bezug*
Modifikator		MO	Modifikator eines Bezugsworts, bspw. eine Praepositionalphrase
Nebensatz	NEB		Verbindet das Verb eines Nebensatzes mit dem uebergeordneten Wort
Negation		NG	

Tabelle 8: Obige Grafik enthält Teil 1 der Dependenz Relationstabelle der Tiger und TueBa-D/Z Treebank, entnommen aus Smith et. Al. [18] für Tiger, sowie Foth et. Al. [7] für TueBa-D/Z.

Dependenz Relation Bezeichnung	**TueBa -D/Z (Malt, ParZu)**	**Tiger (Mate)**	**Beschreibung/Beispiel**
nominales Kernelement		NK	Markiert saemtliche Nomen, die nicht mittels DA, GMOD, OA, OC, OG, OP, SB markiert sind
Zahlenkomponente		NMC	
Zweites Subjekt eines Satzes	NP2		*Das Wetter war schoen, und der [Himmel] blau.*
Akkusativobjekt	OBJA	OA	
Zweites Akkusativobjekt	OBJA2	OA	Fuer Verben, die zwei Akkusative verlangen (lehren, kosten, nennen)
Objektsatz	OBJC	OC	*Ich will wissen, was passiert [ist]*
Dativobjekt	OBJD	DA	*vertrau [mir]*
Genitivobjekt	OBJG	OG	*des bedarf [dessen] nicht*
Objektinfinitiv	OBJI		*ich schlage vor, zu [rasten]*
Praepositionalobjekt	OBJP	OP	*wir greifen [auf] Standardmethoden zurueck*
Parenthese	PAR	PAR	*das, [sagte] er, kann nicht sein*
Eingeschraenkter Partikel	PART		*von Norden [aus] ; nicht [zu] glauben*
Praedikat		PD	
Phrasengenitiv		PG	*eine Schaar [von] Engeln*
Platzhalter		PH	
morphologisches Partikel		PM	*zum Abendessen [zu] bleiben*
Komplement einer Praeposition	PN		Letztes Wort einer Praepositionalphrase; im [Winter]
Teil eines Nomens		PNC	*[Baron] Alexander*
Praepositionalphrase	PP		Leitet Praepositionalphrase ein; [im] Winter
Praedikative Ergaenzung	PRED		*Wir waren [Helden].*
Satzwurzel oder unidentifizierbares Wort	ROOT	ROOT	*Wir [waren] Helden.*
Relativsatz	REL	RC	Verbindet das Verb eines Relativsatzes mit dem uebergeordneten Wort
Wiederholtes Element		RE	
direkte Rede		RS	
Fragment eines Satzes	S		
Subjekt	SUBJ	SB	*[Wir] waren Helden.*
Subjektsatz	SUBJC		*[Gewinnen] ist alles.*
passiviziertes Subjekt (PP)		SBP	*[von] der Natur*
Subjekt oder Praedikat		SP	
Teil einer Einheit		UC	
Vokativ		VO	
Anrede	VOK		*[Sir], es ist angerichtet*
Konjunktionslose Zeitangabe	ZEIT		*Ich fahre jeden [Tag] Fahrrad.*

Tabelle 9: Obige Grafik enthält Teil 2 der Dependenz Relationstabelle der Tiger und TueBa-D/Z Treebank, entnommen aus Smith et. Al. [18] für Tiger, sowie Foth et. Al. [7] für TueBa-D/Z.

Phrasenbezeichnung	Negra	Beschreibung/Beispiel
Superlativphrase mit „am“	**AA**	*Karl lachte [am lautesten]*
Adjektivphrase	**AP**	*die [an sich netten] Melodien*
Adverbialphrase	**AVP**	*[AVP: gar nicht]*
Koordinierte Adpositionen	**CAC**	Verbindung von APPR, APZR, APPO die Zuege [von und nach] Hamburg
Koordinierte Adjektivphrase	**CAP**	*[CAVP: [ADV: nur heute] oder nie]*
Koordinierte Adverbialphrase	**CAVP**	*[CAVP: [ADV: nur heute] oder nie]*
Koordinierte Komplementizer	**CCP**	*[ob und wann] er kommt*
Chunk	**CH**	
Koordinierte Nominalphrase	**CNP**	*[CO: [NP: jeden Tag] oder* *[PP: zumindest am kommenden Freitag]]*
Koordination	**CO**	*[CO: [NP: jeden Tag] oder* *[PP: zumindest am kommenden Freitag]]*
Koordinierte Praepositionalphrase	**CPP**	PP+PP: [CPP: [in der Stadt] und [auf dem Lande]]
koordinierter Satz	**CS**	*[CS: [S: Peter kommt] und [S: Paul geht]]*
Koordinierte Verbphrase	**CVP**	*Er wollte [CVP: [VP: uns besuchen] oder anrufen]*
Koordinierter, mit „zu“ markierter Infinitiv	**CVZ**	*[[zu vergessen] und [zu vergeben]]*
Bestandteil direkter Rede	**DL**	*[[„Lass mich in Ruhe!“] [aergerte sich Peter]]*
Spezifische Einheit (idiosyncratic unit)	**ISU**	
Mehrwoertiges Nomen	**MPN**	Namen: [Karl Schulz]
Multi-Token Adjektiv	**MTA**	*die [MTA: Bad Godesberger] Buerger*
Multi-Token Zahl	**NM**	*[NP: [NM: eine Million] Menschen]*
Nominalphrase	**NP**	*[NP: der alte Mann]*
Adpositonalphrase	**PP**	*[PP: in der,Stadt]* *[PP: meiner Meinung nach]*
Pseudosprache (quasi-language)	**QL**	
Satz	**S**	*[S: Gut, [S: dass du kommst]]*
Verbalphrase	**VP**	*Peter will [VP: uns besuchen]*
Mit „zu“ markierter Infinitiv	**VZ**	*Er versucht, uns [VZ: zu taeuschen]*
Komma	**$,**	,
Satzbeendende Interpunktion	**$.**	. ? ! ; :
sonstige Satzzeichen; satzintern	**$[**	- [ ] ( )

Tabelle 10: Obige Grafik enthält die Phrasentabelle der NEGRA Treebank, entommen Online [22]. Zur gemeinsamen Verwendung mit STTS Part-of-Speech Tags Abbildung (6+7).

9 Tabellenverzeichnis

Tabellenverzeichnis

10 Abbildungsverzeichnis

Abbildungsverzeichnis

11 Quellenverzeichnis

Literatur

[1] Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428, 2013.

[2] Daniel Cer, Marie-Catherine de Marneffe, Dan Jurafsky, and Chris Manning. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*. LREC, 2010.

[3] Jinho D. Choi, Joel Tetreault, and Stent Amanda. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 387–396. Association for Computational Linguistics, 2015.

[4] Alexander Clark, Chris Fox, and Shalom Lappin. *The handbook of computational linguistics and natural language processing*. John Wiley & Sons, 2013.

[5] Michael Collins. Probabilistic context-free grammars (pcfgs). *Lecture Notes*, 2013.

[6] Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, Joao Graca, and Fernando CN Pereira. Frustratingly hard domain adaptation for dependency parsing. In *EMNLP-CoNLL*, pages 1051–1055, 2007.

[7] Kilian A Foth. Eine umfassende constraint-dependenz-grammatik des deutschen. 2006.

[8] Johan Hall, Jens Nilsson, Joakim Nivre, Beáta Megyesi, Mattias Nilsson, and Markus Saers. Single malt or blended? a study in multilingual parser optimization. In *In Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL*, 2007.

[9] Markus Krug. Kalimachos annotation editor, 2016. [Online; accessed 08-February-2016], Available at https://gitlab.informatik.uni-wuerzburg.de/mak28ma/ATHEN.

[10] Sandra Kuebler, Ryan McDonald, and Joakim Nivre. *Dependency Parsing*. Morgan and Claypool, 2009.

[11] Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, pages 915–932. sn, 2007.

[12] Joakim Nivre, Johan Hall, and Jens Nilsson. MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006), May 24-26, 2006, Genoa, Italy*, pages 2216–2219. European Language Resource Association, Paris, 2006.

[13] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics, 2006.

[14] Kenji Sagae and Jun'ichi Tsujii. Dependency parsing and domain adaptation with lr models and parser ensembles. In *EMNLP-CoNLL*, volume 2007, pages 1044–1050, 2007.

[15] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the international conference on new methods in language processing*, volume 12, pages 44–49. Citeseer, 1994.

[16] Rico Sennrich, Gerold Schneider, Martin Volk, and Martin Warin. A new hyprid dependency parser for german.

[17] ACL's Special Interest Group on Natural Language Learning SIGNLL. Conll conference shared tasks website, 2016. [Online; accessed 09-February-2016], Available at http://ifarm.nl/signll/conll.

[18] George Smith. A brief introduction to the tiger treebank, version 1. 2003.

[19] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011.

[20] Christos Stergiou and Dimitrios Siganos. Neural networks. [Online; accessed 04-February-2016].

[21] IMS Stuttgart. Erlauterungen zum stts tagset, 1995. [Online; accessed 15-February-2016], Available at http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc.

[22] uni saarland. List of the phrasal categories used in the negra project, 2006. [Online; accessed 15-February-2016], Available at http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/knoten.html.

[23] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *ACL (1)*, pages 434–443, 2013.