Abteilung Computerlinguistik
Universität Zürich

Sommersemester 2000

Seminar „Syntaxtheorien und computerlinguistische Praxis“
Prof. Dr. Michael Hess, lic. phil. Simon Clematide, lic. phil. Gerold Schneider

Supertagging without tears[1]

Rebecca Schraner
Limmatstr. 209
8005 Zürich
reschraner@yahoo.com

Yvonne Archer
Dreilindenstr. 5
8636 Wald
yvonnearcher@bluewin.ch

September 2000

[1] www.cis.upenn.edu/-mickeyc/stag/stag/stag.html

Inhaltsverzeichnis

1. Einleitung 4

2. Robustes Parsing 5
   2.1 Finite-State-Grammar-based Partial Parsers 5
   2.2 Stochastische Parser 6

3. Lexikalisierte Grammatiken 7
   3.1 Lexikalized Tree-Adjoining Grammar 7
      3.1.1 Inititale Bäume 8
      3.1.2 Auxiliare Bäume 8
      3.1.3 Kompositionsoperationen 9
   3.2 Kerneigenschaften von LTAGs 10
      3.2.1 Bestandteile einer lexikalisierten Grammatik 10
      3.2.2 Extended domain of locality (EDL) 11
      3.2.3 Factoring of recursion from the domain of dependencies (FRD) 12
   3.3 Das XTAG-System 13

4. Was ist Tagging? 16
   4.1 Regelbasiertes Tagging 16
   4.2 Statistisches Tagging 17
   4.3 Anforderungen an einen Tagger 18

5. Supertagging 19
   5.1 Was ist ein Supertag? 19
   5.2 Was ist Supertagging? 21
   5.3 Wie funktioniert der Supertagger? 22

6. Reduzieren von Supertag-Ambiguitäten durch den Gebrauch von strukturellen Informationen 23

6.1 Zulässigkeitsbedingungen 24

7. Modelle zur Reduktion von Ambiguität 26
7.1 Das Unigram-Modell 26
7.1.1 Experimente und Resultate des Unigram-Modells 27
7.2 Das N-gram-Modell 28
7.2.1 Experimente und Resultate des Trigram-Modells 28
7.2.1.1 Experiment I 28
7.2.1.2 Experiment II 29

8. Der Lightweigt dependency Analyzer 30

9. Beispielsätze 33
9.1 Satztypen (Fragesätze, Aussagesätze, Nebensätze) 33
9.2 Unterscheidung Komplement/Adjunkt 37
9.3 Raising-Konstruktionen, Infinitive, Hilfsverben 41

10. Schlusswort 46

Anhang

1. Einleitung

In der vorliegenden Arbeit beschreiben wir in Anlehnung an die Dissertation von Srinivas Bangalore[1] neue Methoden für robustes Parsing, welche die Flexibilität von linguistisch motivierten lexikalischen Beschreibungen mit der Robustheit von statistischen Techniken kombinieren. Srinivas These besagt, dass die Verarbeitung von linguistischen Strukturen lokalisiert werden kann, wenn lexikalische Items mit reichen Beschreibungen, sogenannten Supertags, versehen werden.
Je komplexer diese Beschreibungen sind, desto zahlreicher werden sie und die lokale Ambiguität nimmt zu. Srinivas überprüfte seine These im Kontext der Tree-Adjoining Grammar (LTAG). Die Supertags in LTAG kombinieren Strukturinformationen mit Informationen über die Abhängigkeit zwischen einzelnen Wörtern.

Im 2. Kapitel zeigen wir die Eigenschaften von robustem Parsing auf und betrachten dazu zwei Typen von Parsern. Im darauf folgenden Kapitel werden wir auf die Kerneigenschaften von lexikalisierten Grammatiken eingehen und den X-Tagger als System, welches im LTAG Formalismus implementiert wurde, ansehen. Anschliessend wenden wir uns dem Thema *Tagging* (Kapitel 4) und *Supertagging* (Kapitel 5) zu. Kapitel 6 widmet sich der Reduktion von Supertagambiguitäten.

[1] Srinivas, 1997

2. Robustes Parsing

Die Nützlichkeit des Parsens für natürliche Sprachen ist oft angezweifelt worden, denn es gibt keine Grammatik, die frei auftretende natürlichsprachliche Texte vollständig abdecken kann und es existieren keine Parser, die robust genug sind, um mit dieser Unzulänglichkeit umgehen zu können. Eine weitere Schwierigkeit natürlicher Sprachen sind die zahlreichen Ambiguitäten, welche den Parsingvorgang erheblich verlangsamen.
Srinivas Bangalore widmet sich in seiner Dissertation dem Robusten Parsen und entwickelt neue Methoden, indem er linguistisch motivierte, reiche und komplexe Beschreibungen für partielles Parsen benützt. Robustes Parsing bedeutet, dass unabhängig von der Grammatikalität eines Satzes ein Parse ausgegeben wird. Während der vergangenen Jahre gab es zahlreiche Versuche, robustes Parsen von natürlicher Sprache zu erreichen. Diese Versuche können grob in zwei Kategorien unterteilt werden: **Grammatik-basierte Parser** und **Statistische Parser**. Anschliessend soll ein Überblick über diese zwei Typen von Parsern gegeben werden.

2.1 Finite-State-Grammar-based Partial Parsers

Dieses System benützt Grammatiken, welche als *cascade finite-state regular expressions recognizers* repräsentiert sind. Die regulären Ausdrücke werden normalerweise manuell hinzugefügt. Diese Systeme produzieren einen einzigen Output und vermeiden Ambiguität, indem sie die längste heuristische Übereinstimmung wählen, falls mehr als ein regulärer Ausdruck zu einer gewissen Stelle des Inputstrings passt. Keines dieser Systeme benützt statistische Informationen, um Ambiguität zu reduzieren.
Die folgenden Punkte sind die Hauptgründe für das Scheitern von Robustheit bei solchen grammatischen Parsern: Unvollständige Lexika; inadäquate grammatische Deckung von Systemen; extrem lange Sätze, die unverhältnismässig viel Zeit brauchen, um einen Parse durchzuführen und ungrammatikalische Texte.

2.2 Stochastische Parser

Ein stochastischer Parser versucht **jedem** String eine Struktur zuzuweisen. Die dazu benötigten Regeln entnimmt er grossen Korpora, die manuell hinzugefügte Parses enthalten. Die resultierende Menge von Regeln ist linguistisch nicht transparent und nur schwer modifizierbar.
In den Regeln sind **Wahrscheinlichkeitsinformationen** encodiert. Der Parser nützt diese Wahrscheinlichkeitsinformationen, um lexikalische und strukturelle Ambiguität aufzulösen. So kann das System jedem Input die Struktur mit der grössten Wahrscheinlichkeit zuweisen. Da dieses System **global** optimale Parses zu erreichen versucht, besteht natürlich die Gefahr von **lokal** falschen Parses.

Stochastische Parser sind robuster als grammatik-basierte Parser. Trotzdem weisen sie einige Schwächen auf. Sie benötigen eine riesige Datenmenge, welche natürlich zuerst eingegeben werden muss. Dies ist aber äusserst zeitaufwendig und somit auch teuer.

3. Lexikalisierte Grammatiken

Lexikalisierte Grammatiken sind besonders geeignet für die Spezifikation von natürlichen Sprachen. Dabei spielt das Lexikon eine besonders wichtige Rolle, denn es dient als Schnittstelle, um syntaktische und semantische Informationen miteinander kombinieren zu können. Als repräsentatives Beispiel einer lexikalischen Grammatik gilt die Lexicalized Tree-Adjoining Grammer (LTAG).

3.1 Lexikalized Tree-Adjoining Grammar

Der Formalismus der Tree-Adjoining Grammar (TAG), auf den wir hier nur knapp eingehen werden, wurde 1975 von A. Joshi und M. Takahashi[2] vorgestellt. Die Strukturen einer TAG bestehen aus Bäumen und nicht aus Strings, wie z.B. bei kontextfreien Grammatiken. Der Vorteil von Bäumen gegenüber anderen Grammatiken liegt darin, dass Bäume eine Tiefe haben, in welcher komplexe strukturelle Zusammenhänge dargestellt werden können. Im Vergleich dazu lässt sich mit den Regeln einer kontextfreien Grammatik nur die Tiefe 1 darstellen.

Abb. 1: Vergleich vonTAG-Baum und kontextfreier Regel[3]

Den Kern der TAG Strukturen bilden sogenannte Elementarbäume, die sich in initiale Bäume und auxiliare Bäume unterteilen lassen. Sie unterscheiden sich nach der Rolle, die sie bei der Kompositionsoperation (Kapitel 3.1.3) einnehmen.

[2] Joshi et al. 1975
[3] Tree-Adjoining Grammars mit Unifikation, S. 5

3.1.1 Initiale Bäume

Die initialen Bäume α stellen die Basis der grammatischen Beschreibung dar. Sie können für sich stehen, da sie ein vollständiges, terminales Blattwort besitzen und einfache linguistische Strukturen repräsentieren, wie z.B. Phrasenstrukturen oder simple Sätze. Unter Blattwort wird dabei die Beschriftungen der Blattknoten, von links nach rechts als String gelesen, verstanden.

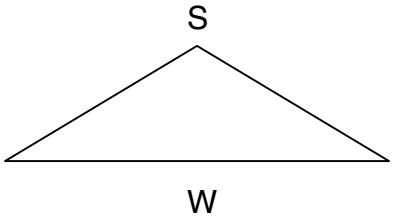
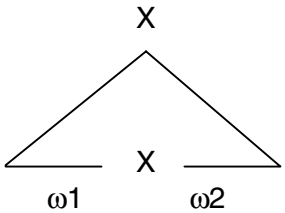


Abb. 2: α = initialer Baum

Der abgebildete Baum zeigt einen S-Typ-Initialbaum, was bedeutet, dass der Baum am Wurzelknoten das Startsymbol S trägt. Jeder gültige Inputstring muss von mindestens einem S-Typ-Initialbaum abgeleitet werden. Es gibt aber auch z.B. X-Typ-Initialbäume, bei welchen der Wurzelknoten mit X beschriftet ist. Bei Initialbäumen ist jeder **innere Knoten** (d.h. Knoten mit ausgehenden Kanten) mit einem Nichtterminal beschriftet. Jeder **Blattknoten** ist mit einem Terminalsympol oder mit ε (dem leeren Wort) etikettiert. Initiale Bäume können nicht in andere eingehängt werden. Das Blattwort w besteht aus Terminalen.

3.1.2 Auxiliare Bäume

Mit auxiliaren Bäumen kann Rekursion beschrieben werden.



$X \varepsilon N$
$\omega 1, \omega 2 \varepsilon T$

Abb. 3: β = auxiliarer Baum

Die Beschriftungen des Fuss- und des Wurzelknotens müssen identisch sein, damit bei der Adjunktion der unter dem Fussknoten liegende Teilbaum nicht vom Gesamtbaum getrennt wird. Der **Wurzelknoten** eines auxiliaren Baumes ist mit einem Nichtterminal beschriftet. Jeder **innere Knoten** ist mit einem Nichtterminal beschriftet. Alle **Blattknoten** sind mit einem Nichterminal- oder einem Terminalsymbol beschriftet. Der Fussknoten dient zur Adjunktion, alle anderen Nichtterminalsymbole sind für die Substitution mit ↓ markiert. Das Blattwort eines auxiliaren Baumes muss **mindestens ein** Terminal enthalten.

3.1.3 Kompositionsoperationen

Elementare Bäume werden durch **Adjunktion**, also **Ineinander-Einhängen**, oder **Substitution** miteinander verknüpft. Initiale Bäume können nicht ineinander adjungiert werden. Auxiliare Bäume können ineinander oder in initiale Bäume eingehängt werden.
Bei der Adjunktion wird ein Fussknoten X in einem initialen oder durch Adjunktion bereits modifizierten Baum α durch einen auxiliaren Baum β mit Wurzel und Fussknoten X ersetzt.

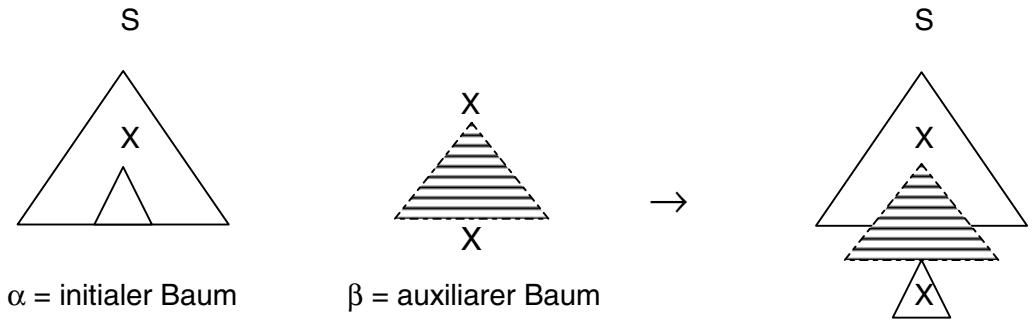


Abb. 4: Adjunktion von Baum β in Knoten X im Baum α

Bei der Substitution wird ein elementarer Baum an den **Substitutionsknoten** eines anderen elementaren Baumes gehängt. Dafür muss ein Knoten eines elementaren

Baumes mit ↓ für die Substitution gekennzeichnet sein. Dieser Knoten wird durch einen anderen elementaren Baum ersetzt, dessen Wurzel-Beschriftung der des durch ↓ gekennzeichneten Fussknotens entspricht.

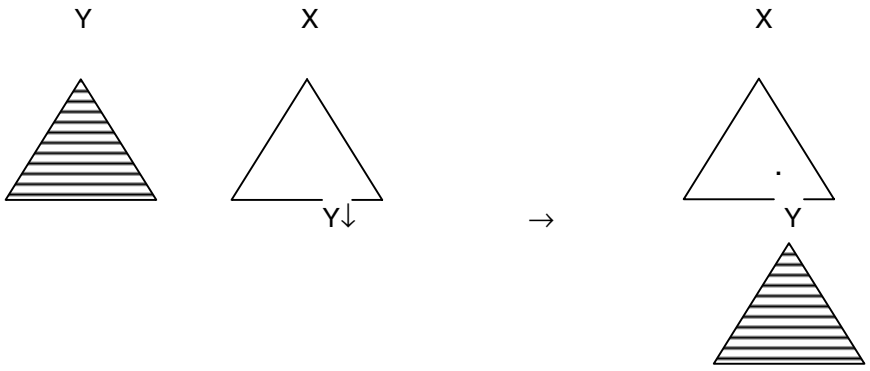


Abb. 5: Substitution des Fussknotens Y durch den elementaren Baum Y

Dank diesem Ineinander-Einhängen eignen sich TAGs besonders zur Beschreibung von natürlichsprachlichen Strukturen. Rekursivität lässt sich durch wiederholtes Ineinandereinsetzen erreichen. Durch Adjunktion lässt sich Erweiterbarkeit, wie z.B. das Einfügen eines Relativsatzes, darstellen.
In LTAG produziert ein Parse zwei Strukturen, nämlich einen Ableitungsbaum und einen abgeleiteten Baum. Die Ableitungsbäume geben darüber Auskunft, wie ein abgeleiteter Baum konstruiert wurde.

3.2 Kerneigenschaften von LTAGs

Zu den Kerneigenschaften einer LTAG zählen: Lexikalisierung, Extended Domain of Locality (EDL) und Factoring of Recursion from Domain of Dependency (FRD).

3.2.1 Bestandteile einer lexikalisierten Grammatik

Eine Grammatik ist lexikalisch, wenn sie sich aus den folgenden Bestandteilen zusammensetzt:

- Einer endlichen Menge von elementaren Strukturen, wie z.B. Strings oder Bäume, wobei jede Struktur in einem lexikalischen Item ankern muss.
- Lexikalischen Items, jedes mit mindestens einer Elementarstruktur der Grammatik verbunden.
- Einem endlichen Set von Operationen, um diese Strukturen miteinander zu kombinieren.

Diese Eigenschaften sind linguistisch entscheidend, weil sie einen direkten Bezug zwischen dem Lexikon und den syntaktischen Strukturen herstellen. In lexikalischen Grammatiken sind die elementaren Strukturen jedes lexikalischen Items nur im Lexikon aufgezeichnet, es gibt keine unabhängige Grammatik.

3.2.2 Extended domain of locality (EDL)

- Jede Elementarstruktur muss alle (und darf nur diese) Argumente des Ankers aufweisen. Anker ist die Bezeichnung für das Terminalsymbol, welches für das lexikalische Element steht.
- Für jede syntaktische Umgebung, in welcher ein lexikalisches Item erscheinen kann, muss die Grammatik eine Elementarstruktur enthalten.

Die folgende Abbildung dient zur Erklärung des ersten Punktes von EDL und zeigt die Elementarstruktur für das Verb *seems* im Satz: *John seems to like Mary*. Im Baum erscheinen nur die Elemente, welche als Argument für den Anker (*seems*) dienen. Deshalb wird, z.B. die NP *John* nicht mit diesem Baum erfasst.

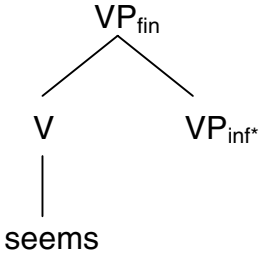


Abb. 6: Elementarbaum des Verbs

Der Anker kann seinen Argumenten syntaktische und semantische Bedingungen auferlegen, weil diese eben in der gleichen Struktur erscheinen. Daher werden alle Argumente, welche in der gleichen Struktur erscheinen, als lokal bezeichnet.

LTAG unterscheidet sich von andern Grammatikformalismen durch den zweiten Punkt von EDL. Jeder Elementarbaum stellt die lineare Ordnung der Argumente des Ankers in einem bestimmten syntaktischen Umfeld dar.

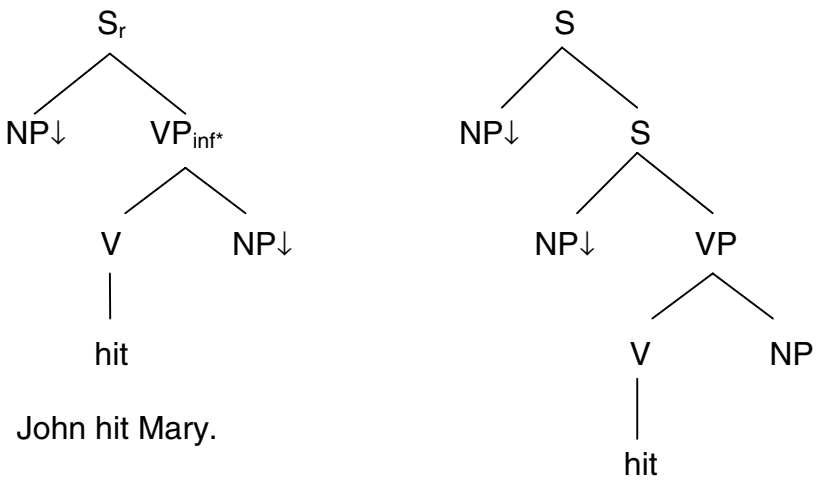


Abb. 7: Elementarbäume des Verbs hit

In Abbildung 7 wird das transitive Verb *hit* mit zwei verschiedenen Kontexten dargestellt. Der linke Baum repräsentiert einen transitiven Aussagesatz, der rechte Baum einen Interrogativsatz mit einem interrogativen Pronominaladverb an der Spitze.
Indem verlangt wird, dass die konstruierten Strukturen „minimal“ sind, folgt die dritte Eigenschaft von LTAG, welche als Folge von EDL zu verstehen ist.

3.2.3 Factoring of recursion from the domain of dependencies (FRD)

Elementarbäume definieren den Bereich für Abhängigkeiten, wie z.B. Übereinstimmung (agreement), Subkategorisierung und Filler-gap Abhängigkeiten. Auxiliare

Bäume, welche durch Adjunktion zu Elementrbäumen werden, zeigen diese Abhängigkeiten über lange Distanzen auf. Rekursion wird in LTAGs durch wiederholte Adjunktion von Auxiliarbäumen und Elementarbäumen erreicht. Die zweite Kerneigenschaft von LTAGs besagt nun, dass Rekursion aus dem Bereich der Abhängigkeiten ausgeschieden wird.
Weil die Merkmalstrukturen einer Grammatik mit EDL und FRD extrem simpel sind und weil der Bereich gross genug für Übereinstimmung, Subkategorisierung und Filler-gap Abhängigkeiten ist, enthalten Merkmalstrukturen keine Rekursion. Sie werden zu typisierten Terms reduziert, welche durch Unifikation kombiniert werden können.

3.3 Das XTAG-System

Die Entwicklung des XTAG-Systems begann 1987. XTAG wurde im lexicalized Tree-Adjoining Grammatikformalismus implementiert und besitzt eine umfangreiche englische Grammatik. Wie oben beschrieben, wird bei einer lexikalisierten TAG jedes einzelne Wort durch mindestens einen (meistens sind es mehrere) Strukturbaum repräsentiert. Der Vorteil eines solchen Formalismus liegt darin, dass der Parsingvorgang erst beginnt, nachdem eine Selektion der Strukturbäume durch die Wörter des Inputstrings vorgenommen wurde. Im XTAG-System wird diese Selektion in verschiedene Schritte aufgeteilt. Bei jedem Schritt wird die Auswahl an Bäumen reduziert und die Ambiguität somit vermindert.

XTAG besteht einerseits aus sprachunabhängigen und andererseits aus sprachabhängigen Komponenten. Zu den sprachunabhängigen Komponenten zählen: ein LTAG-Parser, eine X-Windows-Oberfläche und Instandhaltungs-Tools. Die sprachabhängigen Komponenten bestehen aus einer lexikalischen Struktur und Baumstrukturen für mehrere Sprachen, einer Morphologie-Datenbank und einem POS-Tagger fürs Englische. Dank der X-Windows-Oberfläche sind die morphologische und die syntaktische Datenbank sehr benutzerfreundlich. Es ist möglich, die Dakenbanken mit Informationen zu ergänzen, Einträge zu löschen oder Updates vorzunehmen.

XTAG ist nicht auf das Englische beschränkt. Grammatiken für Chinesisch, Hindi und Koreanisch werden entwickelt.

Funktionsweise XTAG-System:

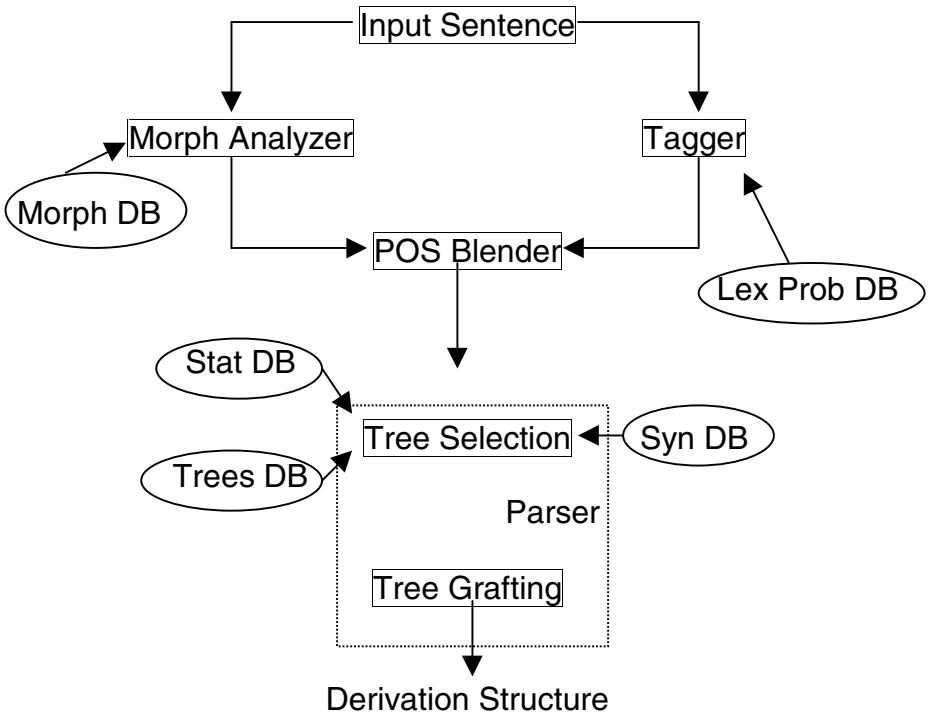


Abb. 8: X-Tagger

Ein eingegebener Satz durchläuft zuerst eine Morphologieanalyse und den POS-Tagger. Bei der Morphologieanalyse wird jedes flektierte Wort in seinen Stamm abgebildet und zusammen mit POS und Flexionskennzeichen in den Ankerknoten (anchor node) des resultierenden Baumes eingefügt. Die Morphologiedatenbank wurde ursprünglich aus Daten aus dem Collins English Dictionary und aus dem Oxford Advanced Learner’s Dictionary erstellt und laufend manuell ergänzt, zur Zeit enthält sie ca. 317.000 flektierte Items. Beim POS Tagger handelt es sich um einen *Wall Street Journal*-trainierten Trigram Tagger, der nur die n-besten POS Sequenzen ausgibt. Dank der Lexical Probabilities-Datenbank verringert sich die Zeit des Parsens um durchschnittlich 93%.

Der POS Blender kombiniert die von der Morphologieanalyse enthaltenen Informationen mit denen des Taggers, indem er den Output des POS Taggers als Filter für den Output der Morphologiedatenbank benützt und gibt anschliessend wiederum nur die n-besten Informationen an den Parser weiter, wobei n für eine natürliche Zahl steht. Dies ist eine spezifische Eigenschaft des X-Taggers. Der Parser des XTAG gliedert sich in zwei Phasen, welche Tree Selection und Tree Grafting genannt werden. Die Trees Datenbank enthält 566 Bäume, welche in 40 Baumfamilien (welche Subkategorisierungsframes enthalten) und 62 individuelle Bäume unterteilt sind. Mit Hilfe der statistischen und der syntaktischen Datenbank werden die lexikalischen Items mit den passenden Bäumen und Baumfamilien aufgrund von Subkategorisierungsinformationen verbunden und an Tree Grafting weitergegeben. Nachdem nun durch Selektion ein Set von Bäumen gefunden wurde, benützt XTAG links-nach-rechts Parsing Algorithmen, um alle Ableitungen des Satzes zu finden.

Die XTAG-Grammatik wurde gebraucht, um Sätze aus Texten u.a. von IBM Manuals und Wall Street Journals zu parsen. Der daraus resultierende XTAG-Korpus enthält Ableitungsbäume, welche über die Konstruktion der abgeleiteten Bäume Auskunft geben.

4. Was ist Tagging?

Statistische Tagger haben die Aufgabe, den Suchraum der Wortartenzuweisung einzuschränken, die Ambiguität zu verkleinern und unbekannte Wörter leichter zu erkennen.

POS-Tagger werden eingesetzt bei Sprachsynthese, Spracherkennung, Information Retrieval, Bedeutungsdisambiguierung und in der Lexikographie.

4.1. Regelbasiertes Tagging

Part-of-speech-Tagging ist die **Zuweisung eines eindeutigen Wortartsymbols** an eine Wortform im Kontext. Dafür ist es vielleicht hilfreich zu wissen, dass die Sprache in Token und Types eingeteilt werden kann. Ein Token ist **jede** in einem Text vorkommende Wortform. Types sind alle in einem Text vorkommenden **unterschiedlichen** Wortformen.

Bsp.: Die Frau jagt die Katze die Treppe hinunter.

Der Beispielsatz hat 8 Token ohne das Satzendzeichen, jedoch nur 6 Types (Gross-/Kleinschreibung nicht berücksichtigt). Wenn man diesen Satz einem POS-Tagger eingibt, erhält man folgende Ausgabe:

Die	ARTDEF	Bestimmter Artikel
Frau	NN	Substantiv (Nomen)
jagt	PDAT	Attributiv gebrauchtes Demonstrativpronomen
die	ARTDEF	Bestimmter Artikel
Katze	NN	Substantiv (Nomen)
die	ARTDEF	Bestimmter Artikel
Treppe	NN	Substantiv (Nomen)
hinunter	ADJA	Attributiv gebrauchtes Adjektiv
.	$.	Satzendezeichen

Abb. 9 Ausgabe des Brill-Taggers

Der POS-Tagger hat jedem Type ein Wortartsymbol zugewiesen, leider nicht jedem das Richtige... „jagt“ wird kaum ein Demonstrativpronomen sein!

Im Deutschen gibt es einige wenige Types, die häufig vorkommen und ganz viele Types, die relativ selten sind.
Die häufigsten Types sind Funktionswörter, vor allem Determiner und Präpositionen. Das liegt daran, dass sie keine Synonyme haben und syntaxspezifisch auftreten. Anstelle von ***der*** *Hut* kann sicher ***dieser*** *Hut* gesagt werden, aber dabei verändert sich der Artikel in ein Demonstrativpronomen, d.h. der bestimmte Determiner hat kein Synonym und wird syntaxspezifisch eingesetzt.
Im Deutschen steht das Type „die“ an erster Stelle, gefolgt von „der“, „und“ und „in“. Das erste Adverb, „nicht“, folgt auf dem achten, das erste Substantiv, „Zeit“, auf dem 90. Rang. „Machen“ ist das erste Vollverb, es ist auf Platz 127 zu finden[4].
Die Zuweisung eines eindeutigen Wortartensymbols folgt auf die morphologische Analyse oder ist selbst lexikonbasiert. Dabei besteht die Hauptarbeit des Taggers darin, zu entscheiden, welchen Tag er dem Token eines Types zuweisen soll. Das Tagging kann entweder statistisch, regelbasiert oder gemischt ablaufen.
Regelbasiertes Tagging hat den Vorteil, dass weniger Information zu verwalten ist. Ausserdem sind die Regeln übersichtlich. Der Tagger kann leicht von Hand verändert werden.

4.2. Statistisches Tagging

Statistisches Tagging bedeutet, dass ein Korpus benützt wird, um den Tagger zu trainieren und dass davon ausgegangen wird, dass die Wahrscheinlichkeit der Aufeinanderfolge von Wortarten unterschiedlich ist. Die Wahrscheinlichkeit der Wortübergänge wird berechnet und über mehrere Wörter hinweg die maximale Wahrscheinlichkeit der Übergänge ermittelt. Der Tagger sucht den wahrscheinlichsten Weg.
Grundlage für das statistische Tagging ist das Hidden Markov Modell. Es basiert auf den Markov-Ketten. Das sind endliche Automaten, bei denen jede Kante mit einer Wahrscheinlichkeit versehen ist. Den Wahrscheinlichkeiten entsprechend sucht der Tagger den besten Weg von Wort zu Wort.
Als Beispiel schauen wir die Nominalphrase „die auf der Bank sitzende Frau“ an. „Die“ kann Relativpronomen, Artikel oder Demonstrativpronomen sein. Für „auf“ ist

eine Zuweisung als Präposition oder Präfix möglich. Relativpronomen, Artikel oder Demonstrativpronomen wären mögliche Wortarten für „der“, „Bank“, „sitzende“ und „Frau“ sind nicht mehrdeutig, haben dementsprechend nur einen zugewiesenen Tag.

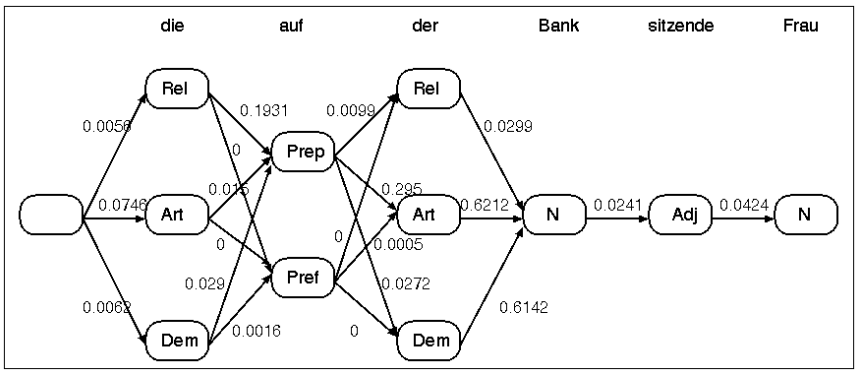


Abb. 10 Bsp. Die auf der Bank sitzende Frau

Aufgrund der angegebenen Wahrscheinlichkeiten für die Übergänge wird sich der POS-Tagger für Artikel, Präposition und Artikel entscheiden.

Statistische Tagger erreichen eine Genauigkeit von 94-97%. Schwierigkeiten bereiten den Taggern unbekannte Wörter, also Wortformen, die im Trainingskorpus nicht vorkamen, und weite Abhängigkeiten im Satz, die über das Bigram- bzw. das Trigramfenster hinausgehen. Auch Aufzählungen, die keine vollständigen Sätze bilden, erschweren dem Tagger das Leben, sofern sie nicht im Korpus vorkommen.

4.1 Anforderungen an einen Tagger

Von einem Tagger wird **Robustheit** erwartet, d.h. er kann einen beliebigen Input verarbeiten, also auch unbekannte Wörter oder Sonderzeichen. Auch **Effizienz** ist wichtig, der Tagger muss schnell arbeiten, wenn er interaktiv verwendet wird. Er soll nur mit geringer Fehlerrate arbeiten (**Genauigkeit**). Unter **Anpassbarkeit** versteht man, dass der Tagger an besondere Anforderungen eines Texttyps angepasst

[4] Meier, H.: Deutsche Sprachstatistik. Hildesheim: Georg Olms. 1964 (nicht mehr ganz aktuell!)

werden kann. Ebenfalls gefordert ist **Wiederverwertbarkeit**: Der Tagger soll leicht für neue Aufgabengebiete eingesetzt werden können.

5. Supertagging

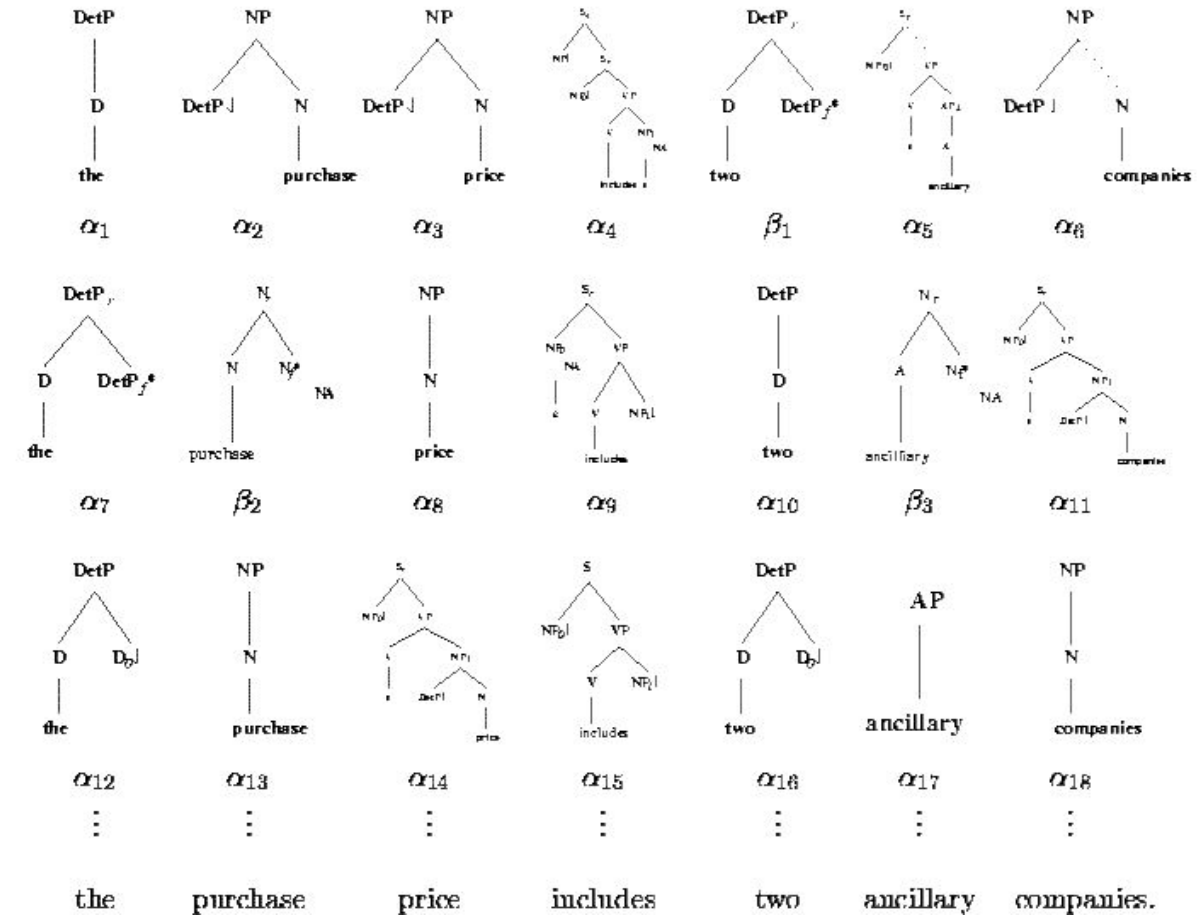
5.1 Was ist ein Supertag?

Supertags sind lexikalische Items mit reicher Beschreibung. Wo der POS-Tagger einfach Wortarten zugewiesen hat, zeigt ein Supertag die möglichen Baumstrukturen, in welchen ein lexikalisches Item auftreten kann. Supertags fordern komplexe Beschränkungen im lokalen Kontext. Informationen wie Subkategorisierung und Übereinstimmung (Agreement) sind in den Supertags enthalten.

Jedes lexikalische Item ist zusammengehängt mit ebenso vielen Supertags, also Bäumchen, wie die Anzahl der verschiedenen Kontexte ist, in der das Item vorkommen kann. Jedes Item ist also mit mindestens einem *elementary tree* assoziiert. Es kann aber auch mehrere elementare Strukturen haben.

Ein Beispiel für elementare Bäume ist die folgende Darstellung, welche die möglichen Supertags für jedes Item des Satzes *the purchase price includes two ancillary companies* aufzeigt:

Abb. 11


DetP
D
the
NP
DetP↓
N
purchase
NP
DetP↓
N
price
DetP
D
two
NP
DetP↓
N
companies
α1
α2
α3
α4
β1
α5
α6
DetP
D
the
N
N*
NA
purchase
NP
N
price
DetP
D
two
AP
ancillary
NP
N
companies
α7
β2
α8
α9
α10
β3
α11
DetP
D
D0↓
the
NP
N
purchase
S
NP↓
VP
V
NP↓
includes
DetP
D
D0↓
two
AP
ancillary
NP
N
companies
α12
α13
α14
α15
α16
α17
α18
the purchase price includes two ancillary companies.

Es gibt **genau einen** Supertag für jede syntaktische Umgebung, in der ein Wort erscheint. Obwohl also ein Wort mit vielen Supertags assoziiert ist, kann es in einem vollständig geparsten Satz nur genau ein Supertag haben.
Supertags beinhalten keine morphologischen Informationen über ein Wort, z.B. *number* oder *tense*. Diese sind in den Attributwertpaaren in der LTAG (Lexicalized Tree-Adjoining Grammar) repräsentiert. Aber Supertags unterscheiden zum Beispiel die beiden ‚to‘ in einem Satz wie *I have to go to New York.*
Supertags enthalten jedoch Dependenzinformationen, d.h. sie lokalisieren Abhängigkeiten und auch Fernabhängigkeiten. Diese Informationen können dazu genützt werden, um die Verteilung von Distanzen zwischen bekannten Supertags und ihren abhängigen Supertags aufzuzeigen.
Der Supertag kombiniert in LTAG sowohl Phrasen-Struktur-Informationen als auch Abhängigkeitsinformationen in **einer** Darstellung. Er verkörpert also die **lexikalische Abhängigkeit, die semantischen und die syntaktischen Beschränkungen** in einer einzigen, einheitlichen Darstellung. Durch diese vielen Informationen kann die Ambiguität vermindert werden.[5]
Um den geeignetsten Supertag für ein Wort zu finden, können lokale statistische Informationen in Form von N-gram-Modellen, die auf der Verteilung von Supertags im LTAG-Korpus basieren, verwendet werden.

5.2 Was ist Supertagging?

Beim Supertagging wird jedes Wort mit einem einzigen Supertag assoziiert. Um die Abhängigkeiten zwischen den Wörtern des eingegebenen Satzes festzustellen, werden die Abhängigkeitsbedingungen, die in den Supertags festgehalten sind, genutzt. Ein *lightweight dependency analyzer* (LDA, robustes Teilparsing)[6], der heuristikbasierte Algorithmen berechnet, produziert mit den strukturellen Informationen und den Abhängigkeitsbedingungen *Dependencylinks*, die nicht notwendigerweise den ganzen Satz umfassen, sondern auch Satzfragmente oder spontane Äusserungen in der wirklichen Welt sein können. Er versucht flachere Strukturen zu bilden.

[5] siehe Kapitel 6
[6] siehe Kapitel 7

Supertagging basiert auf der *Lexicalized Tree-Adjoining Grammar*. Es kann also mit Substitutionen und Adjunktionen umgehen. Die Substitutionsknoten in den Supertags dienen als Platz, wo die Argumente und Komplemente des Supertag**ankers** gespeichert sind. Der Fussknoten eines Supertags ist mit einem Wort gefüllt, das vom Supertagger näher bestimmt worden ist.

In der Standard-POS-Desambiguierung könnte das Supertagging auch von einem Parser übernommen werden. Indem jedoch die POS-Desambiguierung herausgenommen und **vor** das direkte Parsing geschoben wird, wird die Aufgabe des Parsers einfacher. Er kann schneller arbeiten, da bereits vor dem Parsing etwa 50% der Supertags mit Hilfe des Supertaggers eliminiert werden.

Supertagging ist ein „Fast-Parsing“ (*Almost-Parsing*) in dem Sinne, dass der Parser (LDA) nur noch die vom Supertagger gefundenen individuellen Strukturen miteinander verlinken muss, um einen vollständigen Parse zu bekommen.

Das Supertagging gebraucht statistische Methoden. Seine Treffsicherheit und Vollständigkeit hängen vom Material ab, das zum Trainieren des Korpus‘ verwendet wurde.

Beim Supertagging werden Wortarten zu Phrasen zusammengehängt, um schneller Parseergebnisse zu erhalten. Sowohl Determiner und Nomen werden als eine einzige Phrase betrachtet, als auch Determiner, Adjektiv und Nomen oder Adjektiv und Nomen. Als Beispiel seien erwähnt „the postman“(auch a postman, this dog), „a little lamb“ oder „long hair“.

5.3 Wie funktioniert der Supertagger?

Der Supertagger benützt das N-gram-Modell, um die besten möglichen Supertags auszuwählen.

Der Supertagger arbeitet in zwei Schritten. Im ersten Schritt schaut der Supertagger im Lexikon nach und wählt alle Supertags aus, die mit jedem Wort im Satz zusammen auftreten können. Um wahrscheinliche Supertags für ein Wort zu finden, können lokale statistische Informationen in Form von N-gram-Modellen, die auf der Verteilung von Supertags im LTAG-Korpus basieren, verwendet werden. (Der LTAG-Korpus basiert auf dem Wall Street Journal- Korpus und wird laufend erweitert.) Das

N-gram-Modell des Supertaggers beinhaltet also die Suche im Lexikon und das Zuweisen der Wahrscheinlichkeiten.

Im zweiten Schritt durchsucht der Lightweight dependency Analyzer im Sinne eines Parsers das Gitter der ausgewählten Supertags und versucht sie durch Substitutions- und Adjunktionsoperationen so zu kombinieren, dass er eine den ganzen Input-String umfassende Derivation erhält.
Am Ende des zweiten Schrittes hat der LDA nicht nur den eingegebenen Satz geparst, sondern auch ein kleines Set von Supertags, gewöhnlich **einen** Supertag, zu jedem Wort assoziiert.
Der Parser versagt aber total, wenn ein Wort der Eingabe vom Supertagger falsch getagged wurde. Dieses Problem könnte allerdings umgangen werden, wenn die Menge der Supertags auf n-best supertags für jedes Wort ausgedehnt werden würde.
Das Erkennen von Phrasen ist ein zusätzliches Modul, das eingesetzt werden kann. Einige Anwendungen aber benötigen nicht unbedingt die komplete Abhängigkeits-struktur des LDA.

6. Reduktion von Supertag-Ambiguitäten durch den Gebrauch von strukturellen Informationen

In einer lexikalisierten Grammatik wie LTAG ist jedes lexikalische Item mit mindestens einem Supertag verbunden. Voneinander abhängige Elemente werden zusammengehängt und besitzen dann eine gemeinsame Struktur. Abhängigkeiten, auch solche über lange Distanzen, können so lokalisiert werden. Als Folge dieser Lokalisierung sind die meisten lexikalischen Items mit mehreren Supertags versehen. Um Ambiguität zu reduzieren, verteilen Supertags einem syntaktischen Umfeld sogenannte Zulässigkeitsbedingungen. Die folgenden Bedingungen haben die Aufgabe, die Zulässigkeit eines syntaktischen Umfeldes für einen Supertag zu prüfen.

6.1 Zulässigkeitsbedingungen

- **Spannweite eines Supertags:**

Mit der Spannweite wird die **minimale Anzahl** lexikalischer Items angegeben, die mit einem Supertag abgedeckt wird. Wenn nun der Inputstring weniger Wörter enthält als im Supertag fixiert sind, dann ist ein Parse unmöglich. So können Supertags, welche auf Spannweite-Beschränkungen basieren, reduziert werden. Wenn also z.B. der Supertag die Bedingung enthält, ein Inputstring müsse mindestens 5 lexikalische Items enthalten, so würde der Inputstring *„wake up“* nicht berücksichtigt werden.

- **Rechts (links) Spannweite-Beschränkungen:**

Die zweite Beschränkung ist eine Spezialform der ersten. Ein lexikalisches Item wird als Anker des Baumes bezeichnet. Der elementare Baum liefert eine komplexe Beschreibung des Ankers.
Wenn die Spannweite eines Supertags rechts (links) des Ankers grösser ist als die Länge des Strings rechts (links) vom Wort, welches den Supertag ankert, dann kann der Supertag für keinen Parse gebraucht werden.

- **Lexikalische Items im Supertag:**

Ein Supertag kann eliminiert werden, wenn das Terminal(symbol), welches an der Grenze des Supertags auftaucht, nicht im Input String auftaucht. Bsp: Supertags mit dem lexikalischen Item by, welches englische Passivkonstruktionen kennzeichnet, werden während des Parsens eines *aktiven* Satzes eliminiert.

Die beschriebenen Bedingungen dienen dazu, Supertags, welche die verlangten Merkmale im Kontext des Inputstrings nicht aufweisen, zu eliminieren. So werden z.B. Supertags, welche ein wh-Wort und eine NP verlangen eliminiert, wenn der Inputstring kein Wort beginnend mit wh- enthält.

Die folgende Tafel zeigt, dass sich solche Filter beim Reduzieren von Supertag-Ambiguität als sehr effizient erweisen.

System	Getaggte Wörter	∅ Anzahl Supertags/Wort
ohne strukt. Beschränkungen	48 783	47.0
mit strukt. Beschränkungen	48 783	25.0

Abb. 12: Supertag-Ambiguität mit und ohne den Gebrauch von strukturellen Beschränkungen.[7]

Abb. 13: Vergleich der Anzahl von Supertags mit und ohne Filter für

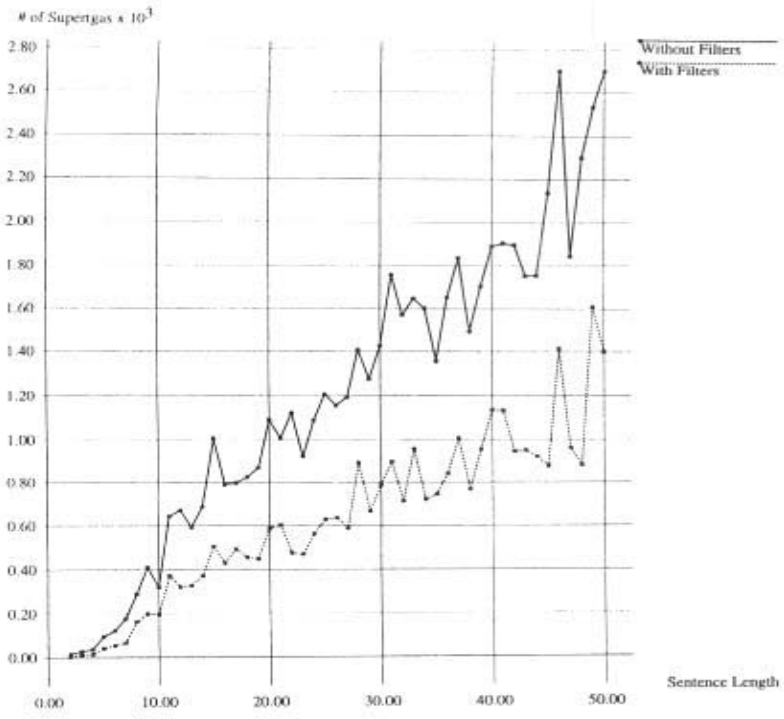


Figure 3
Comparison of number of supertags with and without filtering for sentences of length 2 to 50 words.

Sätze mit einer Länge von 2 bis 50 Wörter

[7]Wall Street Journal Section 20 of the Penn Treebank

Bei einem gegebenem Satz kann unter Anwendung von strukturellen Beschränkungen ungefähr die Hälfte aller Supertags eliminiert werden. Die Phase des Filterns findet schon vor Beginn des Parsens statt. Durch die Reduktion von Supertags verringert sich die Dauer des Parsingvorgangs. Ohne den Gebrauch dieser Filter ist der Parser im XTAG-System äusserst langsam, denn die Anzahl der Ambiguitäten ist enorm.

7. Modelle zur Reduktion von Ambiguität

7.1 Das Unigram-Modell

Das **Unigram-Modell** ist eine Methode zu Disambiguierung: Die Anwendung von struktureller Information zum Ausfiltern von unmöglichen Supertags reduziert zwar die Ambiguitäten, eliminiert sie jedoch nicht vollständig. Es besteht die Möglichkeit, die Disambiguierung eines Wortes besser bewältigen zu können, wenn seine bevorzugten Tags in eine **Rangreihenfolge** gebracht werden. Die Häufigkeit, mit der ein bestimmter Supertag mit einem bestimmten Wort auftritt, ist ein direktes Mass für die lexikalische Vorliebe genau dieses Tags. Der beliebteste Tag, mit dem ein Wort verknüpft wird im Trainingskorpus, ist dem Unigram-Modell entsprechend als **der** Supertag ausgewählt. Das Unigram-Modell wird folgendermassen beschrieben: die Wahrscheinlichkeit P für den korrekten Supertag ist gleich der Häufigkeit, mit der ein Tag und ein Wort zusammen auftreten, dividiert durch die Häufigkeit des Wortes im Trainingskorpus.
[P (Tag | Wort)=Häufigkeit (Tag, Wort):Häufigkeit (Wortvorkommen im Korpus)]

7.1.1 Experimente und Resultate des Unigram-Modells

Zum Trainieren und Testen des **Unigram-Modells** wurden zwei verschiedene Datensätze benützt. Der erste Datensatz wurde aus Parses des Wall Street Journal Korpus, des IBM-Manual Korpus und des ATIS Korpus gewonnen. Dabei wurde die *wide-coverage English Grammar* verwendet, die als Teil des XTAG-Systems entwickelt wurde. Diese Parses werden nachfolgend als XTAG-Parses bezeichnet.
Für den zweiten und grösseren Datensatz wurden die Penn Treebank-Parses von Wall Street Journal Sätzen konvertiert.
Den Wörtern wurden zuerst unter Verwendung eines konventionellen Taggers ein Part-of-speech-Tag zugewiesen. Danach wurde ihnen mit Hilfe des Unigram-Modells ein Supertag zugewiesen. Ein Wort wird dann als mit dem korrekten Supertag ausgestattet angesehen, wenn es sich um den gleichen Supertag handelt, wie in einem korrekten Parse des Satzes. Die Resultate dieses Experiments zeigen sich in der folgenden Tabelle:

Datensatz	Anzahl Wörter im Trainingsset	Anzahl Wörter im Testset	Top n Supertags	Erfolg in %
XTAG Parses	8 000	3 000	n = 1 n = 2 n = 3	73.4% 80.2% 80.8 %
Konvertierte Penn Treebank Parses	1 000 000	47 000	n = 1 n = 2 n = 3	77.2% 87.0% 91.5%

Abb. 14: Resultate des Unigram-Supertag Modells

Obwohl das Unigram-Modell mit Supertagging deutlich schlechter abschneidet als das Unigramm-Modell mit POS-Tagging, ist dessen Leistung viel höher als erwartet, denn die Anzahl Supertags ist viel grösser als die Anzahl POS-Tags. Einer der Gründe für die hohe Leistung ist, dass die häufigsten Supertags für die häufigsten Wörter, wie Artikel, Nomen, Hilfsverben meist auch der korrekte Supertag ist. Das Zurückgreifen auf Part-of-speech ist also beim Supertaggen von unbekannten

Wörtern, welches meistens Nomen sind, oft von Nutzen. Die meisten Fehler macht das Unigramm-Modell bei Verben, Präpositionen und Nomen.

7.2 Das N-gram-Modell

Im Unigram-Modell wird ein Wort mit dem Supertag versehen, welcher gesamthaft gesehen am häufigsten gewählt wird. Der Kontext, in dem das Wort erscheint, wird dabei ausser acht gelassen. Eine alternative Methode dazu bietet das N-gram-Modell, welches den Kontext beim Zuweisen eines Supertags miteinbezieht.
Das N-gram-Modell berücksichtigt nämlich die kontextuellen **Abhängigkeits-Wahrscheinlichkeiten** zwischen den einzelnen Supertags innerhalb eines Fensters mit n-Wörtern. Der Output ist die Supertag Sequenz mit der **grössten Wahrscheinlichkeit**.

7.2.1 Experimente und Resultate des Trigram-Modells

Die Leistungsfähigkeit des Trigram-Modell wurde anhand verschiedener Korpora getestet. Es waren dies der Wall Street Journals (WSJ) Korpus, der IBM Manual Korpus und der ATIS Korpus.

7.2.1.1 Experiment I

Im Experiment I wurde die Leistungsfähigkeit des Supertaggers im Wall Street Korpus getestet. Dazu wurden die Daten von XTAG-Parses und von konvertierten Penn Treebank-Parses benützt.

Data Set	Anzahl Wörter im Trainingsset	Modell	Anzahl Wörter im Testset	% korrekt
XTAG Parses	8 000	Unigram Trigram	3 000 3 000	73.4 % 86.0 %
Konvertierte	2 000	Unigram	47 000	75.3 %

Penn Treebank Parses		Trigram	47 000	90.0 %
	1 000 000	Unigram	47 000	77.2%
		Trigram	47 000	92.2 %

Abb. 15: Leistungsfähigkeit des Supertaggers beim WSJ Korpus

Die Datenmenge der **XTAG-Parses** wurde aufgeteilt in Trainings- und Testmaterial. Das Trainingsmaterial umfasst 8‘000 Wörter, das Testmaterial 3‘000 Wörter. Die Daten aus der **Penn Treebank** wurden für zwei Experimente, die sich in der Grösse des Trainingskorpus unterscheiden, gebraucht. Die Anzahl Wörter im Testset betrug 47‘000 und blieb bei beiden Versuchen konstant. Einmal enthielt das Trainingsset 200‘000 Wörter, das 2. Mal 1‘000‘000 Wörter. Für diese Tests wurden gesamthaft 300 verschiedene Supertags benötigt.

7.2.1.2 Experiment II

Für das Experiment II wurde der IBM Korpus und der ATIS Korpus benützt.

Korpus	Anzahl Wörter im Trainingsset	Modell	Anzahl Wörter im Testset	% korrekt
IBM Manual	14 000	Unigram	1 000	77.8 %
		Trigram	1 000	90.3 %
ATIS	1 500	Unigram	400	85.7 %
		Trigram	400	93.8 %

Abb. 16: Leistungsfähigkeit des Supertaggers überprüft im IBM Manual Korpus und im ATIS Korpus

Abb. 16 zeigt die Leistung des Supertaggers im IBM Manual Korpus im Vergleich zum ATIS Korpus. Auffallend dabei ist die geringe Anzahl Wörter im Trainingsset des ATIS Korpus. Auch das Testset enthält relativ wenig Wörter.
Gesamthaft gesehen schnitt der Supertagger im ATIS Korpus trotz des kleinen Trainingssets am besten ab. Der IBM Manual Korpus liegt vor dem WSJ Korpus, wenn die Grösse des Trainingskorpus‘ berücksichtigt wird.
Das Trigram-Modell für Supertagging ist attraktiv für limitierte Bereiche, denn es erzielt die besten Resultate für relativ kleine Mengen an Trainingsmaterial. Die Leistungsfähigkeit des Supertaggers kann gesteigert werden, indem der Supertagger zum Supertaggen von grossen Trainingskorpora benützt wird, welche anschliessend von Hand korrigiert werden und zum Trainieren von Supertaggern verwendet werden.

8. Der Lightweigt dependency Analyzer

Sobald eine Reihe von Supertags vom Supertagger identifiziert worden ist, beginnt die Arbeit des LDA. Er kreiert die Abhängigkeitsstrukturen, d.h. er durchsucht den Strukturbaum jedes Supertags nach Substitutions- und Fussknoten, welche er mit Supertags von anderen Wörtern füllen kann.
Um Dependenzlinks zwischen den Wörtern eines Satzes herzustellen, werden die Abhängigkeitsforderungen genützt, die in den Supertags verschlüsselt sind. Die Substitutionsknoten und die Fussknoten in den Supertags dienen als Platz, welcher mit den Argumenten des Supertagankers gefüllt werden muss. Ein Substitutionsplatz wird von Ankerkomplementen gefüllt, ein Fussknoten von einem durch Supertag modifizierten Wort. Diese Argumente haben einen Polaritätswert, der ihre Orientierung hinsichtlich des Supertagankers widerspiegelt.
Ebenfalls mit dem Supertag assoziiert ist eine Liste von internen Knoten (Wurzelknoten eingeschlossen), die im Supertag erscheint.
Wenn man die strukturelle Information nun zusammen mit den Argumentsforderungen eines Supertags nützt, kann mit einem Algorithmus eine Methode bereitgestellt werden, um den Satz mit Dependenzlinks zu versehen.

Der Output dieses Algorithmus sieht folgendermassen aus für den Satz *The implicit interior state of the iteration over the hash table entries has dynamic extent*:

Pos	Word	Supertag	Slot req.	Pass 1	Pass 2	Dep Links
0	The	β 1	+NP*	3*	-	3*
1	implicit	β 2	+N*	2*		2*
2	interior	β 2	+N*	3*		3*
3	state	α 2	-	-	-	-
4	of	β 1	-NP* +NP.	3*6.		3*6.
5	the	β 1	+NP*	6*	-	6*
6	iteration	α 2	-	-	-	-
7	over	β 1	-NP* +NP.	6*11.		6*11.
8	the	α 1	+NP*	11*		11*
9	hash	β 3	+N	10*		10*
10	table	β 3	+N	11*		11*
11	entries	α 2	-	-	-	-
12	has	α 3	+NP.-NP.		3.14.	3.14.
13	dynamic	β 2	+N*	14*		14*
14	extent	α 4	-	-	-	-

Die Bedeutung der verschiedenen Kolonnen:

1 Position des Wortes im Input

2 Wörter des Inputs (token)

3 dem Token entsprechender Supertag

4 Kontextbedingungen

5 / 6 geben die 2 Orte für Dependenz an. * und . verweisen auf die Art der Abhängigkeit. * ist modifizierende Relation, . ist Komplement

7 Abhängigkeitslinks

Das Wort in Position 4, *of*, fordert links eine NP, die näher bestimmt wird durch *of* und rechts eine NP als Komplement. *Of* ist von beiden abhängig. Das Komplement selber ist ein Initialbaum und somit unabhängig.

Etwas anders sieht es bei Position 7, *over*, aus. Es hat zwar die gleich nahe Abhängigkeit wie *of*, aber eines seiner abhängigen Links liegt 4 Positionen weiter weg. Dennoch hat der LDA diese lokale Beschränkung und partielle Abhängigkeit erkannt.

Der LDA ist ein heuristik-basierter deterministischer Algorithmus, der die Abhängigkeitslinkages produziert, die nicht unbedingt den ganzen Satz umfassen. Er kann eine Anzahl partielle Linkages produzieren, da er primär von der Notwendigkeit angetrieben wird, lokale Beschränkungen zufriedenzustellen ohne eine einzelne Abhängigkeitsverbindung zu machen, die den ganzen Input umfasst. Das trägt tatsächlich zur Robustheit des LDA bei und verspricht ein nützliches Tool zum Parsen von Teilsätzen zu sein, die ja oft in sprachlichen Äusserungen vorkommen.

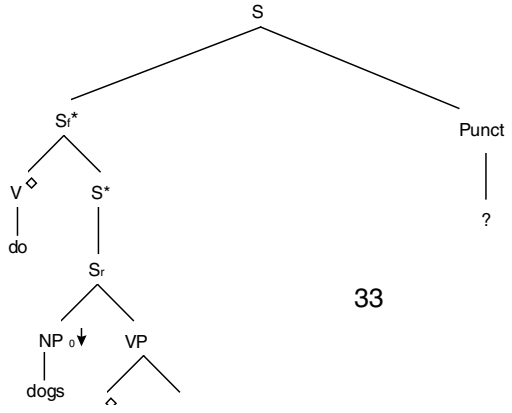
9. Beispielsätze

9.1 Satztypen (Fragesätze, Aussagesätze, Nebensätze)

Do dogs bite?

0 Do B_Vs 2*
1 dogs A_NXN
2 bite A_nx0Vnx1 1. ??.
3 ? B_sPU 2*
...EOS...

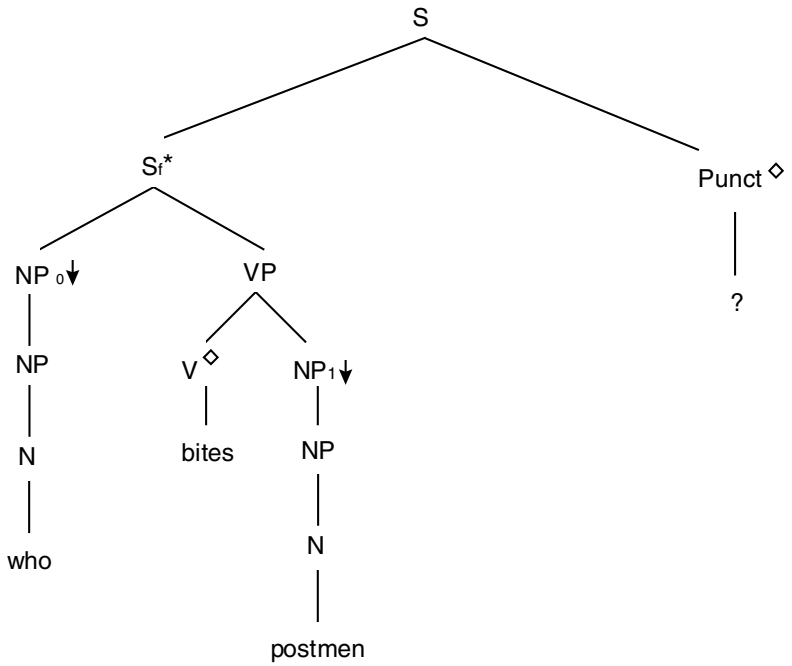
A und B geben jeweils an, ob es sich um Initial- oder Auxiliarbäume handelt. Bei Knoten 0 bedeutet also der Grossbuchstabe B, dass es sich um einen Auxiliarbaum handelt. Vs steht für auxiliare Inversion. Im Gegensatz zu XTAG erkennt der Supertagger die Inversion. 2* zeigt die Fussknotenabhängigkeit des Auxiliarbaumes an, *do* hängt also vom Hauptverb *bite* ab.
Knoten 1 *dogs* ist ein Alphabaum, also ein Initialbaum. NX besagt, dass eine NP als Wurzelknoten (*root node*) auftritt. Das zweite N bedeutet, dass die NP als Anker ein Nomen hat.
Bite bei Knoten 2 ist ebenfalls ein Initialbaum (A). nx0Vnx1 steht für transitives Verb. Die 1. verweist auf eine interne Substitution durch Knoten 1. Die beiden Fragezeichen deuten an, dass dem transitiven Verb ein Objekt fehlt. Im Strukturbaum wird also ein Knoten nicht mit einem lexikalischen Item gefüllt sein. Das Fragezeichen ist ein Auxiliarbaum, s steht für Satz und PU ist die Interpunktion. Die 2* verweist wieder auf Knoten 2, d. h. das Satzendzeichen soll vom transitiven Verb fussknotenabhängig sein.
Am Ende jeder Ausgabe erscheint EOS = End Of Sentence.



Who bites postmen?

O Who A_NXN
1 bites A_nx0Vnx1 0. 2.
2 postmen A_NXN
3 ? B_sPU 1*
... EOS...

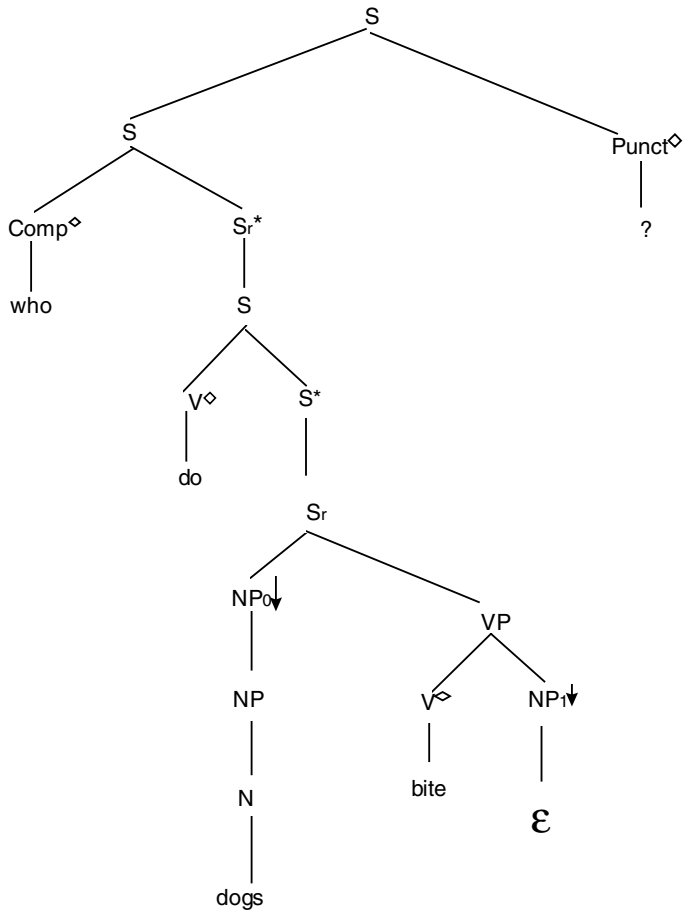
Das *Who* wird nicht als Wh- Konstruktion erkannt, es wird als Nominalphrase analysiert. Bei Knoten 1 verweisen die beiden Zahlen am Ende der Zeile auf Substitution: Knoten 0 und Knoten 2 werden im Elementarbaum von *bites* zwei interne Knoten durch ihre Strukturen ersetzen.
Der Verweis bei Knoten 3 zeigt, dass der Auxiliarbaum des Satzendzeichens fussknotenabhängig ist von Knoten 1.



Who do dogs bite?

0 Who B_COMPs 1*
1 do B_Vs 3*
2 dogs A_NXN
3 bite A_nx0Vnx1 2. ??
4 ? B_sPU 3*
...EOS...

B_COMPs bei Knoten 0 ist ein Satzkomplement. Auch hier wird also die Wh-Konstruktion nicht erkannt. 1* verweist auf die Fussknotenabhängigkeit mit dem Knoten 1.Der String von *bite* erklärt, dass durch den Baum in Knoten 2 eine Substitution in einem internen Knoten stattfindet und dass dem Verb ein Objekt fehlt. Es nimmt nicht *who* als Komplement.



Who does this dog belong to?

0 Who B_COMPs 1*
1 does B_Vs 3*
2 this _ dog A_NXN
3 belong A_nx0Vpx1 2. 4.
4 to A_PXPnx ??
5 ? B_sPU 3*
...EOS...

nx0Vpx1 bezeichnet ein transitives Verb mit Präpositionalphrase (=px) als direktes Objekt (=1).
Der String für „to“ bedeutet Präpositionalphrase. Die Fragezeichen deuten auf das fehlende Objekt hin.
Für diese Ausgabe konnten wir keinen Baum zeichnen, weil uns das System den Elementarbaum A_nx0Vpx1 nicht zeigen wollte.

Dogs that bark bite.

0 Dogs A_NXN
1 that B_COMPs 3*
2 bark bite A_NXN
3 . B_sPU
...EOS...

Auffallend ist hier, dass die zwei Verben *bark* und *bite* zu einer Nominalphrase mit einem Nomen als Anker zusammengefasst werden. Die Lexikoneinträge im Korpus zeigen, dass sowohl *bark* als auch *bite* Nomen und Verb sein können. Der Tagger entscheidet sich für die N-N-Kombination, weil er den Relativsatz nicht als solchen erkennt. Es missfällt ihm nicht, dass kein Verb vorkommt.

Durch das Fehlen des Verbes ist es unmöglich, einen vollständigen, zusammenhängenden Baum zu zeichnen. Die Elementarstrukturen bleiben Elementarstrukturen.

If a dog barks it bites.

0 If B_COMPs 2*
1 a_dog A_NXN
2 barks B_nx0Vs1 1. 4*
3 it A_NXG
4 bites A_nx0V 3.
5 . B_sPU 4*
...EOS...

If wird als Satzcomplementizer analysiert. Es substituiert in *barks* einen Knoten. *Barks* ist als Verb mit Satzkomplement beschrieben, und das in Knoten 1 substituiert wird und fussknotenabhängig ist von Knoten 4 *bites*.
It soll ein Initialbaum mit Genitiv als Subjekt sein...
Bites ist ein intransitives Verb, in welchem ein interner Knoten durch den Knoten 3 substituiert wird.
Diese Strings konnten wir nicht zu einem Baum zusammenfügen, weil uns der Elementarbaum von diesem Genitiv als Subjekt fehlte.

9.2 Unterscheidung Komplement/Adjunkt

The student of English with long hair bites the dog.

0 The_student A_NXN
1 of B_nxPnx 0* 2.
2 English A_NXN
3 with B_nxPnx 2* 4.
4 long_hair A_NXN

5 bites A_nx0Vnx1 0. 6.

6 the_dog A_NXN

7 . B_sPU 5*

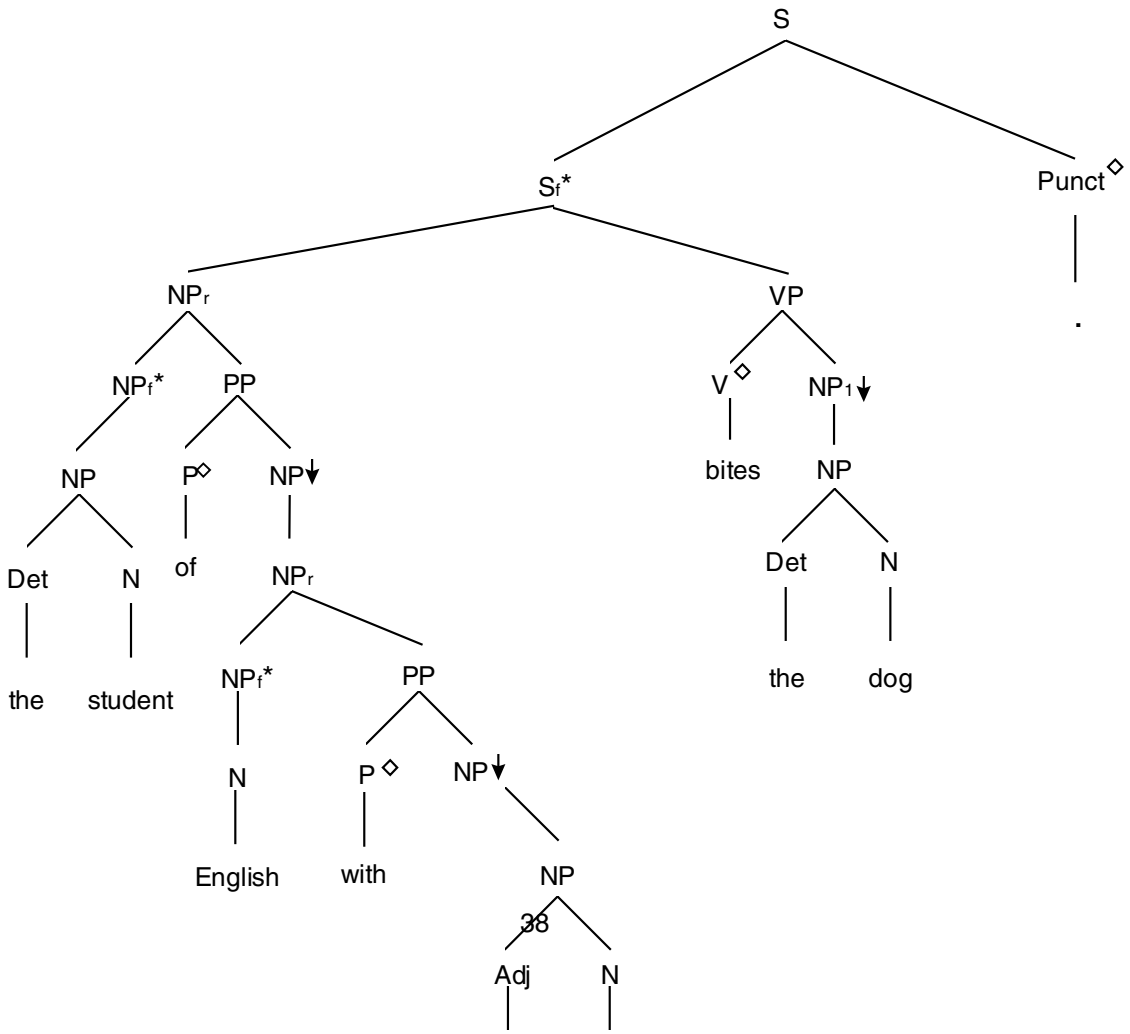
...EOS...

Der String von Knoten 1 bedeutet eine Präposition, die eine Nominalphrase modifiziert. Bei diesem Knoten steht ein Auxiliarbaum (B). Seine Fussknotenabhängigkeit (*) verweist auf Knoten 0. Knoten 1 substituiert in Knoten 2 einen internen Knoten durch den Elementarbaum (2.).

With in Knoten 3 erhält einen String, der auf die Fussknotenabhängigkeit von 2, also English, hindeutet und zeigt auf, dass Knoten 4 substituiert wird.

Nx0Vnx1 , der String von Knoten 5, sagt aus, dass bite ein transitives Verb ist und verweist auf die Knoten 0 und 6, die zusammen mit 5 den Hauptsatz *The student bites the dog* bilden.

Der Supertagger hat in diesem Beispielsatz nicht erkannt, dass *the student with the long hair* zusammengehören müsste, er hat die langen Haare dem *English* zugeschrieben.



The postman gives a bone to the dog every day.

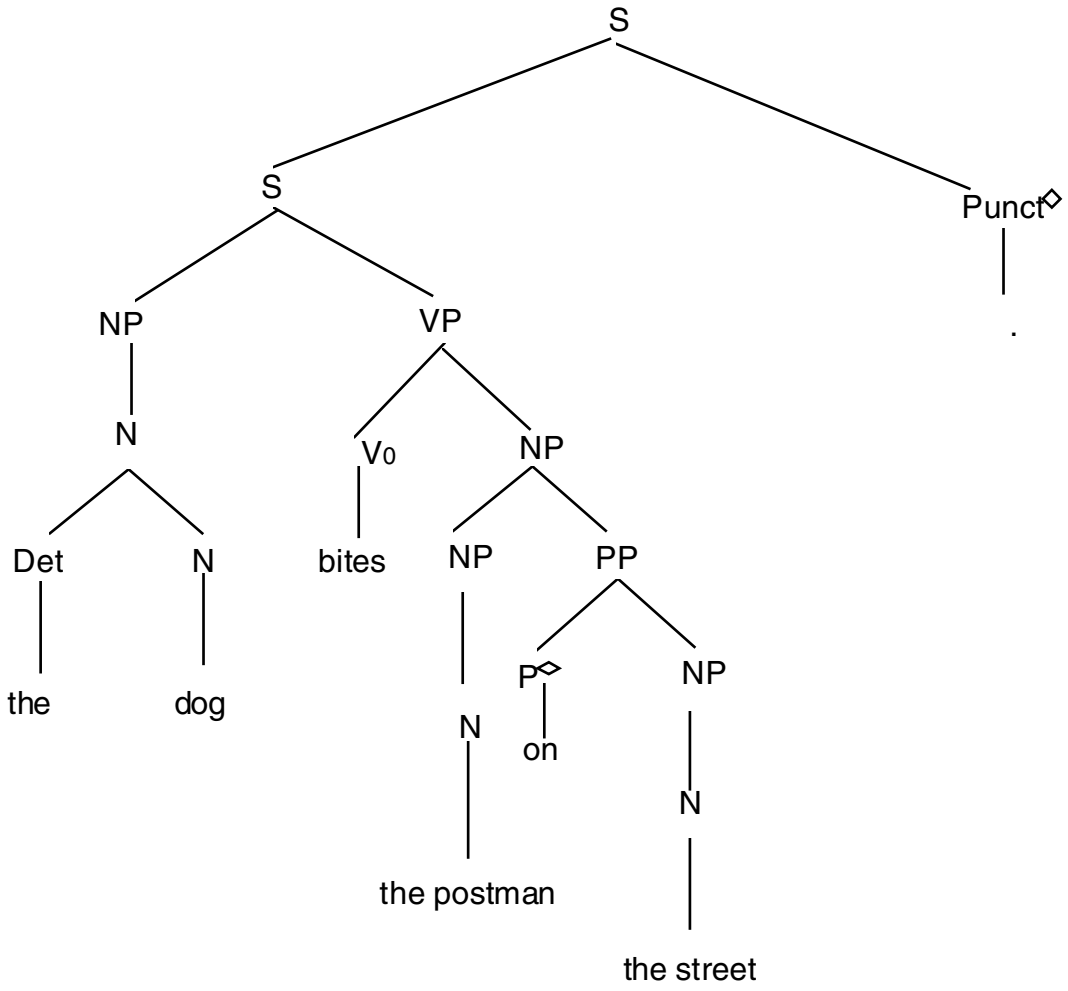
0 The postman A_NXN
1 gives A_nx0Vnx1nx2 0. 2. 4.
2 a_bone A_NXN
3 to B_Vvx 5&
4 the_dog_every A_NXN
5 day B_vxN 1*
6 . B_sPU 1*
...EOS...

gives wird als ditransitives Verb erkannt, es ist ein Initialbaum und indiziert interne Substitutionen durch die Strukturen der Knoten 0, 2 und 4 .
Als ein Auxiliarverb wird *to* eingestuft. Es erhält zudem noch die Ergänzung *5&.* Die Ziffer steht für den entsprechenden Knoten und *&* hat ungefähr die gleiche Bedeutung wie * . Der einzige Unterschied ist, dass *&* auch eine andere Adjunktion an seinem Abhängigkeitsknoten erlauben würde.
Eigenartigerweise schliesst der Supertagger *the dog* mit *every* zusammen, statt *every* mit *day*. *The dog every* wird als initiale Nominalphrase mit Anker Nomen analysiert, *day* als auxiliare Nominalphrase mit Anker Verb.
Durch diese Falschausgabe des Taggers ist ein vollständiges Parsing des Satzes unmöglich. Es gibt nur Satzfragmente, die zusammengesetzt werden können.

The dog bites the postman on the street.

0 The_dog A_NXN
1 bites A_nx0Vnx1 0. 2.
2 the_postman A_NXN
3 on B_nxPnx 2* 4.
4 the_street A_NXN
5 . B_sPU 1*
...EOS...

Der Auxiliarbaum von Knoten 3 mit der Bezeichnung *Noun phrase modifying preposition* und den Hinweisen auf Fussknotenabhängigkeit mit Knoten 2 und Substitution durch Knoten 4 zeigt, dass *the postman* mit *on the street* zusammengehängt wird. Die Präpositionalphrase modifiziert also in diesem Tag die Nominalphrase.

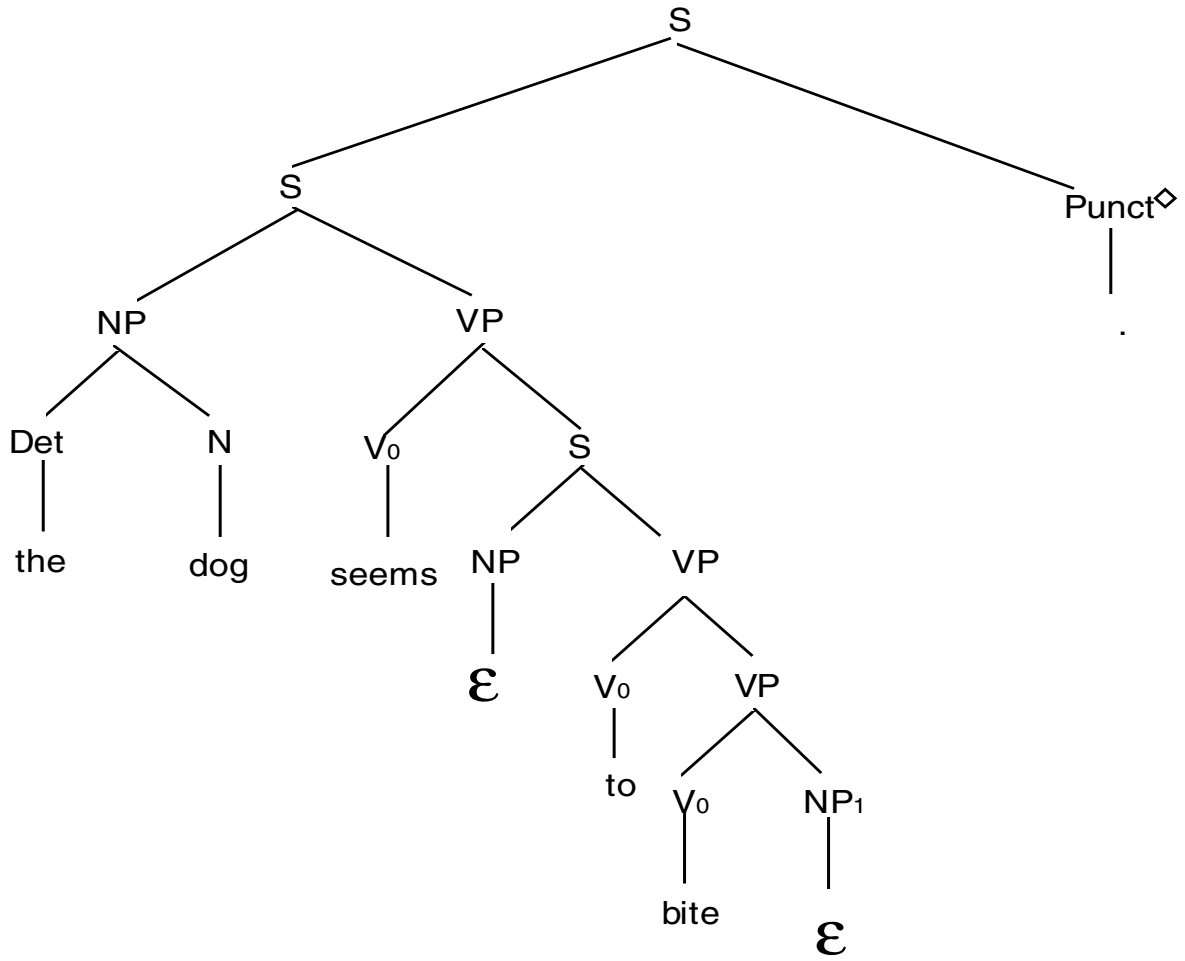


9.3 Raising-Konstruktionen, Infinitive, Hilfsverben

The dog seems to bite.

0 The_dog A_NXN
1 seems B_nx0Vs1 0. 3.
2 to B_Vvx 3*
3 bite A_nx0Vnx1 ??. ??.
4 . B_sPU 3*
...EOS...

seems erhält hier die Zuweisung zu den Verben mit Satzkomplement. Es hat interne Knoten, die durch die Elementarbäume der Knoten 0 und 3 ersetzt werden. Der Tagger kann mit dieser Satzkonstruktion umgehen und einen sinnvollen Parse ausgeben. Die Fragezeichen zeigen, dass *bite* weder Subjekt noch Objekt hat. Im Baum werden das leere Plätze sein.



The dog wants the postman to give him a bone.

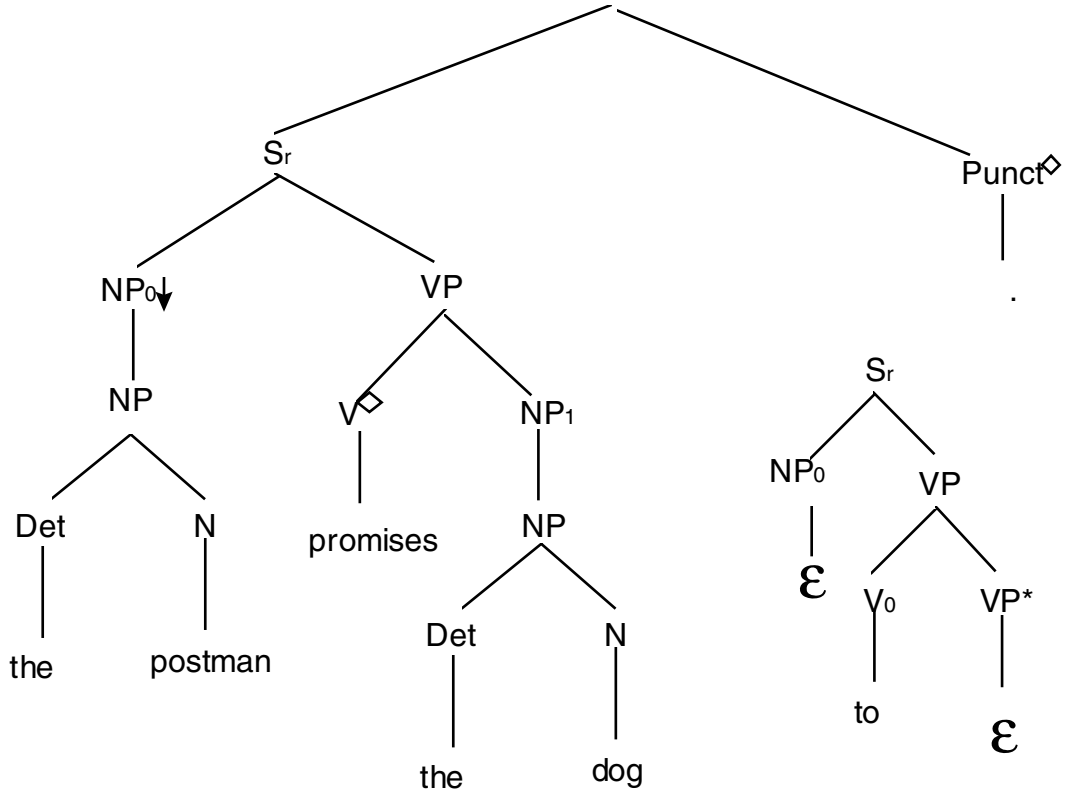
0 The_dog A_NXN
1 wants B_nx0Vs1 0. 4*
2 the_postman A_NXN
3 to B_Vvx 4*
4 give A_nx0Vnx1nx2 2. 5. 6.
5 him A_NXG
6 a_bone A_NXN
7 . B_sPU 4*
...EOS...

wants wird als Verb mit Satzkomplement analysiert. Es hat Fussknotenabhängigkeit von Knoten 4.
Der String von Knoten 4 verweist auf die Substitutionen durch die Strukturen von Knoten 2, 5 und 6.
him ist Subjekt mit Gerundium...

The postman promises the dog to bring a bone.

0 The_postman A_NXN
1 promises A_nx0Vnx1 0. 2.
2 the_dog A_NXN
3 to B_Vvx 4*
4 bring A_nx0Vnx1 ??. 5.
5 a_bone A_NXN
6 . B_sPU 4*
...EOS...

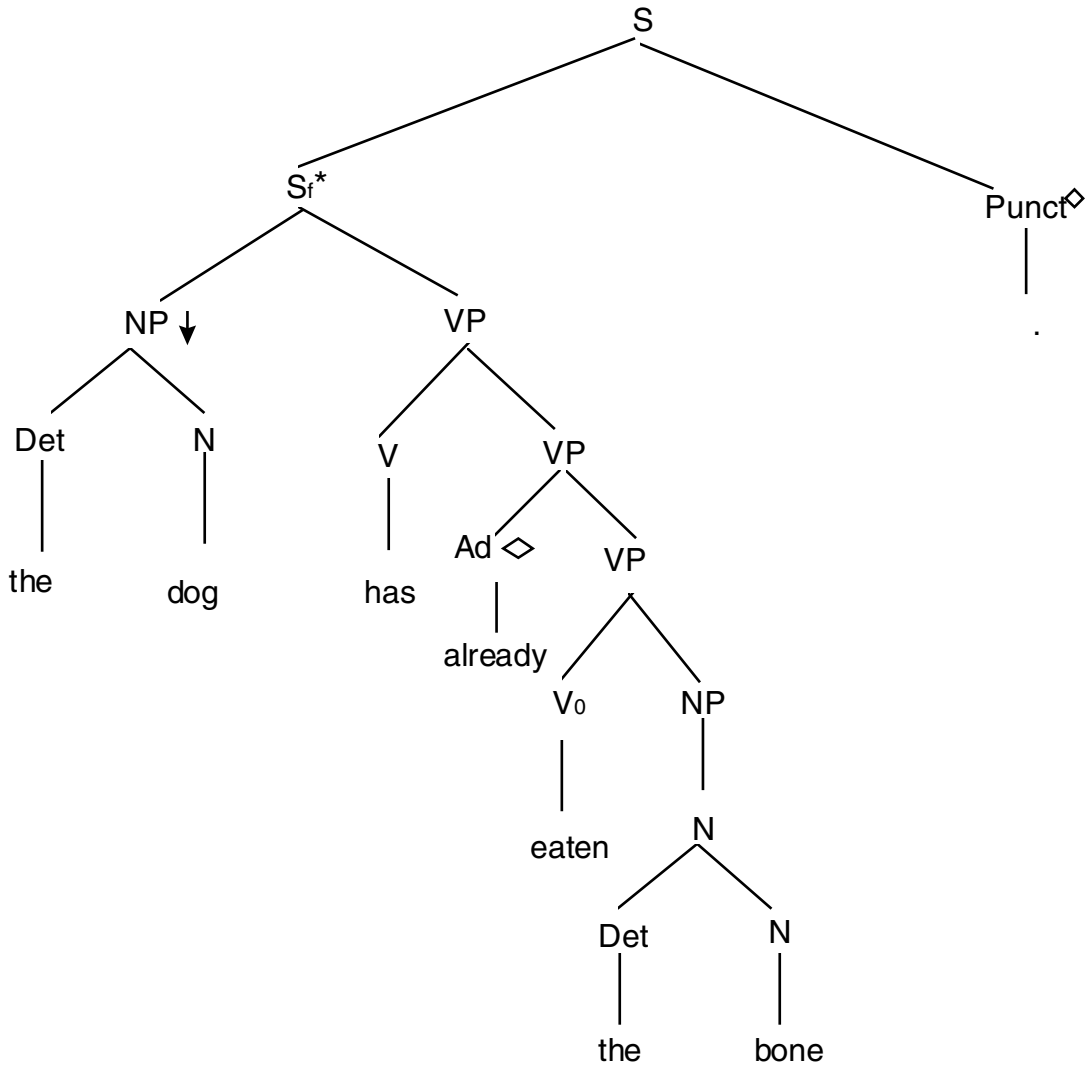
Die Infinitivkonstruktion wird von Supertagger erkannt. Was der Tagger nicht findet, ist ein Item für die Subjektsposition von Knoten 4. Er zeigt dies mit den zwei ?.
Gibt er einen vollständigen Baum aus? Versuche ihn zu zeichnen:



The dog has already eaten the bone.

0 The_dog A_NXN
1 has B_Vvx 2&
2 already B_ARBvx 3*
3 eaten A_nx0Vnx1 0. 4.
4 the_bone A_NXN
5 . sPU 3*

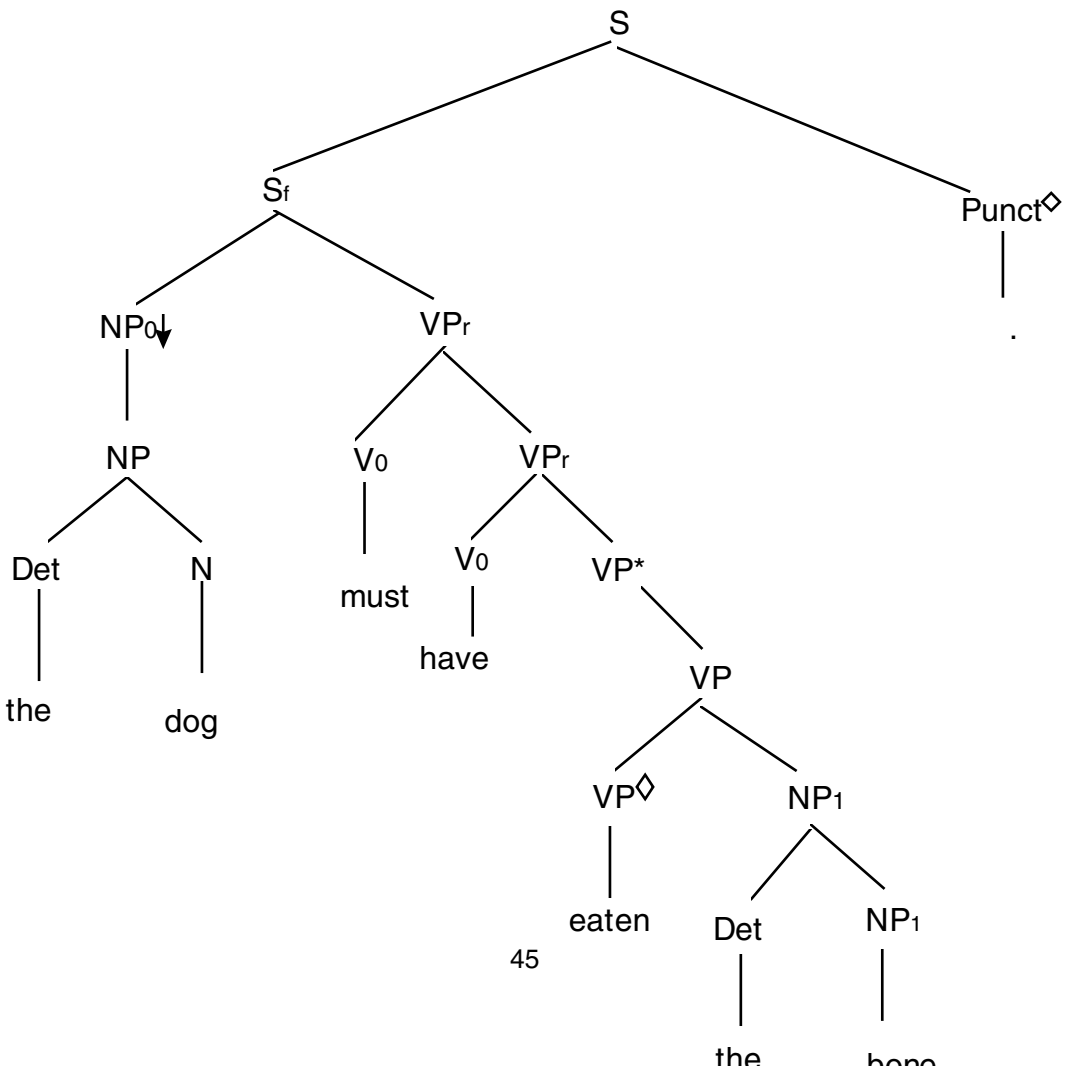
Has in Knoten 1 wird als Hilfsverb gekennzeichnet. Das & sagt aus, dass auch eine andere Adjunktion erlaubt wäre bei Knoten 2. Knoten 2 gilt als Adverb, welches eine Verbalphrase modifiziert.
Interne Knoten des Elementarstrukturbaumes von *eaten* werden durch die Initialbäume von *the dog* und *the bone* ersetzt.



The dog must have eaten the bone.

0 The_dog A_NXN
1 must B_Vvx 2&
2 have B_Vvx 3*
3 eaten A_nx0Vnx1 0. 4.
4 the_bone A_NXN
5 . B_sPU 3*
...EOS...

Must und *have* erkennt der Supertagger als Hilfsverben. Knoten 1 kann in Knoten 2 adjungiert werden und ein interner Knoten von *eaten* wird durch Bäume der Knoten 0 und 4 ersetzt.



10. Schlusswort

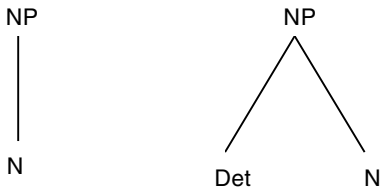
Der Supertagger hat unserer Meinung nach eine hoffnungsvolle Zukunft vor sich. Die Vorteile, die ihn auszeichnen sind zwar nicht immer ganz ersichtlich. Es passieren Fehler beim Taggen, die in der Folge auch das Parsen unmöglich machen. Aber eben: wer macht keine Fehler?
Dennoch macht der Supertagger das Parsing schneller, indem er den Schritt des Fast-Parsings vor dem richtigen Parsen übernimmt und damit den Suchraum für den Parser verkleinert. Mit den verschiedenen Strukturen, die jedes Item hat, geht er gänzlich neue und andere Wege als die POS-Tagger.
Sowohl Rekursion als auch Relativsätze sind für den Supertagger lösbar. Wh-Konstruktionen analysiert er als Nominalphrasen. Obwohl dies kein vollständiges Parsing ist, lassen sich zusammenhängende Strukturen ableiten.
Als Plus kann zudem angeführt werden, dass der Supertagger auch mit andern Grammatikformalismen als XTAG arbeiten kann. Nennen könnte man zum Beispiel HPSG und LFG.

Nachteilig wirken sich die schon erwähnten Fehler beim Taggen aus. Wird ein Item falsch getagged, erhält es einen String, der sich nicht mit andern aus dem Eingabesatz zusammenfügen lässt zu einem sinnvollen Ganzen. Damit ist das korrekte Parsing ausgeschlossen. Eine Schwierigkeit, die sich dem Supertagging in den Weg stellt, ist die Datenbank, übrigens dieselbe des XTAG, in der die meisten Verben auch als Nomen oder umgekehrt geführt werden. Der Supertagger scheitert dadurch manchmal an Eingaben, wie wir es im Beispielsatz „*dogs that bark bite*“ gesehen haben.
Trotz des Versprechens, der Supertagger könne auch Fernabhängigkeiten bewältigen, ist es ihm nicht immer gelungen, lange Eingaben zu einem vollständigen Parse zu analysieren .Oft können die dabei geparsten Satzfragmente nicht zu einem Strukturbaum zusammengestellt werden.
Diese Tatsache kann aber auch ein Vorteil sein, denn in natürlichen Sprachen kommt es oft vor, dass nur Satzfragmente gebraucht werden und nicht in vollständigen Sätzen kommuniziert wird.

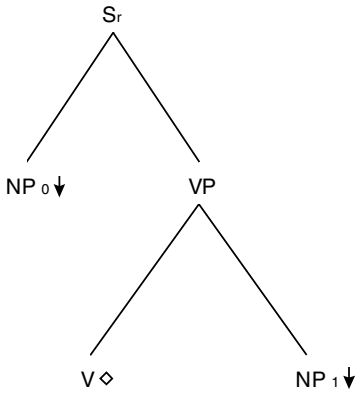
Anhang

Alphabäume

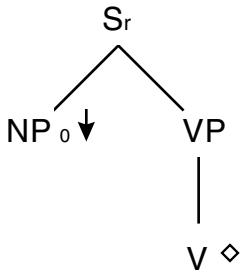
NXN



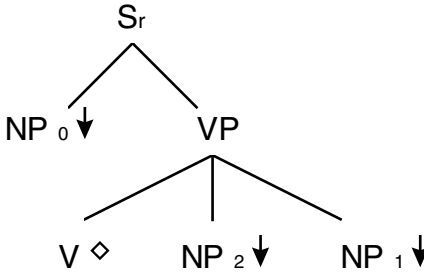
nx0Vnx1



nx0V

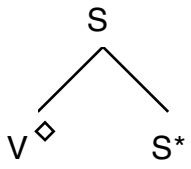


nx0Vnx1nx2

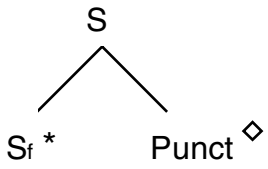


Betabäume

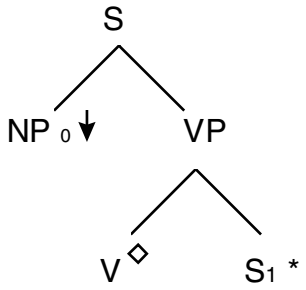
Vs



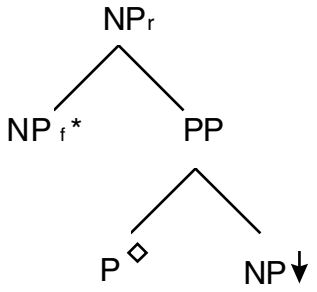
sPU



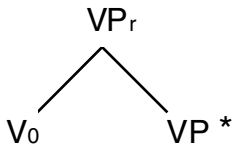
nx0Vs1



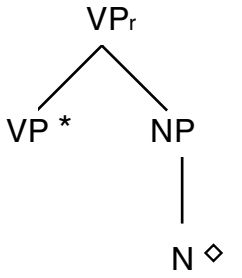
nxPnx



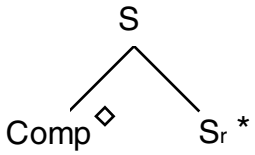
Vvx



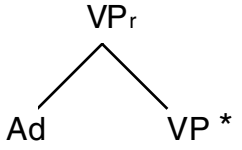
vxN



Comps



ARBvx



Literaturverzeichnis

Buschauer, Béla/ Harbusch, Karin/ Poller, Peter/ Schauder, Anne: Tree-Adjoining Grammars mit Unifikation. Technical Meno. Kaiserslautern / Saarbrücken 1991 (DFKI Publikationen, TM-91-10)

Chandrasekar, R: The Tree-Adjoining Grammar Formalism (TAG). http://www.cis.upenn.edu/-mickeyc/stag/stag/

Doran, Christine / Egedi, Dania / Hockey, Beth Ann / Srinivas, B. / Zaidel, Martin: XTAG System- A Wide Coverage Grammar for English. In: Proceedings of the 15th International Conference on Computational Linguistics (COLING ’94). Kyoto 1994. Volume of August. S. 922-928

Doran, Christine / Hockey, Beth Ann / Sarkar, Anoop / Srinivas, B. /Xia, Fei: Evolution of the XTAG System. Auf: http://www.cis.upenn.edu/ˆxtag

Joshi, Aravind K. / Schabes, Yves: Tree-Adjoining Grammars. In: Rozenberg, Grzegorz / Salomaa, Arto (Hsg.): Handbook of Formal Languages. Volume 3. Beyond Words. Berlin / Heidelberg 1997. S. 69-89

Joshi, Aravind K. / Levy, Leon S. / Takahashi, Masako: Tree Adjunct Grammars. Journal of computer and System Sciences. New York / London 1975. Volume 10:1, S. 136-163.

Srinivas, Bangalore: Complexity of lexical descriptions an its relevance to partial parsing. A dissertation in computer and information science, Philadelphia, 1997. Auf: http:///www. Cis.upenn.edu/~srini/

Srinivas, Bangalore / Joshi, Aravind K.: An Aproach to Almost Parsing. In**:** Computational Linguistics. June 1999. Volume 25, Number 2

The XTAG Research Group: a Lexicalized Tree-Adjoining Grammar for English. Philadelphia, PA, Institute for Research in Cognitive Science, 15. Januar 1999. http://www.cis.upenn.edu/~xtag

Volk, Martin: Formale Grammatiken und Syntaxanalyse. Vorlesung 5. Mai 1998. Auf: http://www.ifi.unizh.ch/CL/volk/LexMorphVorl/Lexikon06.Freq.html