Lehrstuhl Angewandte Informatik IV
Datenbanken und Informationssysteme
Prof. Dr.-Ing. Stefan Jablonski

Institut für Angewandte Informatik
Fakultät für Mathematik, Physik und Informatik
Universität Bayreuth

Bachelorarbeit

Klaus Freiberger

Juni 1, 2019
Version: Final

Universität Bayreuth

Fakultät Mathematik, Physik, Informatik
Institut für Informatik
Lehrstuhl für Angewandte Informatik IV

Rapid Miner NLP Tools: Syntax Parsing

Bachelorarbeit

Klaus Freiberger

1. Reviewer Prof. Dr.-Ing. Stefan Jablonski
Fakultät Mathematik, Physik, Informatik
Universität Bayreuth

2. Reviewer Dr. Lars Ackermann
Fakultät Mathematik, Physik, Informatik
Universität Bayreuth

Supervisors Stefan Jablonski and Lars Ackermann

Juni 1, 2019

Klaus Freiberger

Bachelorarbeit

Rapid Miner NLP Tools: Syntax Parsing, Juni 1, 2019

Reviewers: Prof. Dr.-Ing. Stefan Jablonski and Dr. Lars Ackermann

Supervisors: Stefan Jablonski and Lars Ackermann

Universität Bayreuth

Lehrstuhl für Angewandte Informatik IV

Institut für Informatik

Fakultät Mathematik, Physik, Informatik

Universitätsstrasse 30

95447 Bayreuth

Germany

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Abstract (different language)

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

1 Einleitung

1.1 Motivation and Problem Statement

1.1.1 Some References

[WEB:GNU:GPL:2010; WEB:Miede:2011]

1.2 Thesis Structure

2 Natural Language Processing

In diesem Kapitel wird der theoretische Hintergrund, die Methoden des Natural-Language-Processings, eingeführt. Nach den Grundlagen werden zwei verschiedene Methoden zum Parsen vorgestellt.

2.1 Grundlagen

Betrachtet man einen Satz in einer natürlichen Sprache, die im Rahmen dieser Arbeit auf Englisch festgelegt ist, so kann ein Algorithmus dessen Semantik nicht ohne Weiteres herauslesen. Hierfür bedarf es verschiedener Hilfsstrukturen und zusätzlicher Informationen. Als ersten Schritt bietet es sich an, den einzelnen Wörtern eines Satzes ihre Wortart zuzuordnen. Mit Wortart, alternativ auch Wortklasse oder im Englischen part-of-speech (POS), ist gemeint, wie ein Wort im Satz auftritt. Beispiele für Wortarten sind Nomen, Verb und Adjektiv. In dieser Arbeit wird das Tagset, also die Menge an Wortarten, aus der Penn Treebank verwendet. Siehe hierfür Tabelle 2.1. Der Satz

> My dog also likes eating sausage.

würde mit diesem Tagset also folgendermaßen annotiert werden:

> My/PRP$ dog/NN also/RB likes/VBZ eating/VBG sausage/NN ./.

Wie ein Satz maschinell mit POS-Tags versehen werden kann, wird in dieser Arbeit nicht weiter behandelt. Über diese zusätzliche Notation hinaus kann man erkennen, dass sich in der englischen Sprache oftmals mehrere Wörter als Gruppe oder als eine Komponente innerhalb des Satzes verhalten. So eine Gruppe wäre zum Beispiel die Nominalphrase *My Dog* oder die Verbalphrase *likes eating sausage*. Auch hier wird wieder die Annotation der Penn Treebank verwendet, abgebildet in Tabelle 2.2. Zu diesem Satz an Tags gibt es noch die Erweiterung um relationale Tags. Diese liefern eine Zusatzinformation und werden an die eben vorgestellten angehängt. Zum Beispiel bekommt das Subjekt eines Satzes das Suffix *-SBJ*. Das heißt aus *NP* wird *NP-SBJ*. Dieser Erweiterungssatz wird im Rahmen dieser Arbeit herausgelassen, aber der Vollständigkeit halber erwähnt. In 2.1 wird der Satz mit der syntaktischen

Penn Treebank Part-of-Speech Tags		
Tag	Beschreibung	Beispiel
CC	Koordinierende Konjunktion	*and*
CD	Kardinalzahl	*third*
DT	Artikel	*the*
EX	Existentielles *there*	*there* is
FW	Fremdword	*les*
IN	Präposition, unterordnende Konjunktion	*in*
JJ	Adjektiv	*green*
JJR	Adjektiv, Komparativ	*greener*
JJS	Adjektiv, Superlativ	*greenest*
LS	Listenelement Markierung	*1)*
MD	Modal	*could*
NN	Nomen, singular oder Masse	*table*
NNS	Nomen, plural	*tables*
NNP	Eingenname, singular	*Germany*
NNPS	Eigenname, plural	*Vikings*
PDT	Predeterminer	*both* his children
POS	possesive Endung	*'s*
PRP	Personalpronomen	*me*
PRP$	Possesivpronomen	*my*
RB	Adverb	*extremely*
RBR	Adverb, Komparativ	*better*
RBS	Adverb, Superlativ	*best*
RP	Partikel	*about*
SYM	Symbol	*%*
TO	to	what *to* do
UH	Ausruf	*oops*
VB	Verb, Grundform	*be*
VBD	Verb, Vergangenheitsform	*was*
VBG	Verb, Gerund \Partizip Präsens	*being*
VBN	Verb, Partizip Perfekt	*been*
VBP	Verb, Präsens, nicht 3.Person Singular	*am*
VBZ	Verb, Präsens, 3.Person Singular	*is*
WDT	*wh*-Artikel	*which*
WP	*wh*-Pronomen	*who*
WP$	*wh*-Possesivpronomen	*whose*
WRB	*wh*-Adverb	*be*

Tab. 2.1: Penn Treebank POS Tags

Penn Treebank Syntactic Tags		
Tag	Beschreibung	Beispiel
S	einfacher deklarativer Satz	*There we go.*
SBAR	Satz beginnend mit unterordnender Konjunktion	feels *like we have to move*
SBARQ	Direkte Frage beginnend mit *wh*-Wort oder *wh*-Phrase	*So what's that about?*
SINV	Invertierter deklarativer Satz	neither *am I a pessimist.*
SQ	Invertierte Ja/Nein Frage oder Hauptsatz einer *wh*-Frage	*Will they move on?*
ADJP	Adjektivphrase	*relatively cheap*
ADVP	Adverbphrase	*down here*
CONJP	Konjunktionalphrase	*but also for tissues*
FRAG	Fragment	if *not today*, ...
INTJ	Zwischenruf	*Well*
LST	Listenmarkierung	*1*
NAC	Keine Komponente	via the *Freedom of Information*
NP	Nominalphrase	*the sun*
NX	Markiert Kopf in komplexen NP	fresh *apples and cinnamon*
PP	Präpositionalphrase	*in some way*
PRN	Nebenläufige Phrase	*..., bless his heart, ...*
PRT	Partikel	*up*
QP	Quantifizierende Phrase	*or two a day*
RRC	Reduzierter Relativsatz	titles *not presently in the collection*
UCP	Ungleich Koordinierte Phrasen	She flew *yesterday and on July 4th.*
VP	Verbalphrase	this *is my dog.*
WHADJP	*Wh*-Adjektivphrase	*how great you are*
WHADVP	*Wh*-Adverbphrase	*When* I see it
WHNP	*Wh*-Nominalphrase	*What* they've done
WHPP	*Wh*-Präpositional	*At which*
X	Unbekannt	*The more ..., the less ...*

Tab. 2.2: Penn Treebank Syntaktische Tags

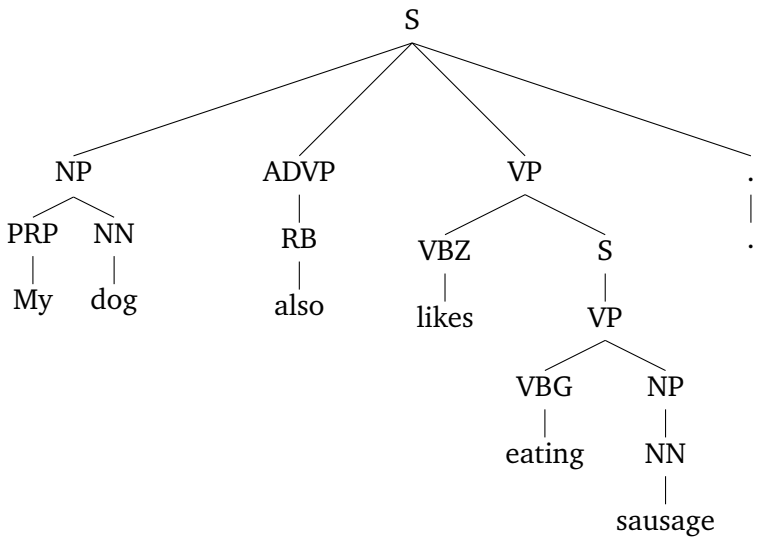

Fig. 2.1: Syntaxbaum zum Satz *My dog also likes eating sausage.*

Annotation dargestellt.

$$
\begin{aligned}
&(S \\
&\quad (NP\ \ (PRP\$\ \ My)\ (NN\ \ dog)) \\
&\quad (ADVP\ \ (RB\ \ also)) \\
&\quad (VP\ \ (VBZ\ \ likes) \\
&\qquad (S \\
&\qquad\quad (VP\ \ (VBGeating) \\
&\qquad\quad (NP\ \ (NN\ \ sausage))))) \\
&\quad (.\ \ .))
\end{aligned}
\tag{2.1}
$$

Alternativ zur geklammerten Lösung kann man dieses Ergebnis auch als Syntaxbaum zeichnen, siehe hierfür Abbildung 2.1.

Wie man am Beispiel erkennen kann, sind, im Gegensatz zu den POS-Tags, auch diverse Verschachtelungen dieser Komponenten möglich. Um diese Anordnungsstruktur innerhalb einer Sprache zu beschreiben, bieten sich kontextfreie Grammatiken an. Diese Art der Grammatik zeichnet sich dadruch aus, dass sich auf der linken Seite der Regeln genau ein Nichtterminalsymbol und auf der rechten Seite können sich beliebig viele Terminale und Nichtterminale befinden. Das Nichtterminal auf der linken Seite hat keine Einschränkung welche Symbole sich um es herum befinden müssen. Es hat also keinen Kontext, daher die Bezeichnung kontextfrei. Für die Verarbeitung unseres Beispielsatzes wurden unter anderem folgende Regeln verwendet:

$S \rightarrow NP\ ADVP\ VP\ .$
$NP \rightarrow PRP\$\ NN$
$PRP\$ \rightarrow My$
$NN \rightarrow dog$

Da ein englischer Satz nicht unbedingt *S* als oberstes Nichtterminal hat, sondern auch *FRAG*, *SBARQ* und andere möglich sind, muss in den Grammatiken ein zusätzliches Startsymbol eingeführt werden. Dieses wird zum Beispiel *ROOT* oder *TOP* genannt. Anhand des vollständigen Regelsatzes der Grammatik einer Sprache, kann man theoretisch jedem grammatikalisch korrekten Satz dieser Sprache seine syntaktische Struktur zuweisen. Es gibt für diverse natürliche Sprachen Sammlungen von annotierten Sätzen, diese werden Treebank oder Korpus genannt. Ein bekannter Korpus der englischen Sprache ist die Penn Treebank, aus welcher auch die hier verwendete Annotation stammt. Dieser Korpus wird vom Linguistic Data Consortium, mit Sitz in der Univertität von Pennsylvania, herausgegeben. Im Rahmen des Penn Treebank Projekts wurden von 1989 bis 1992 über 4,5 Millionen Wörter der Treebank hinzugefügt. Die Texte hierfür stammen zu einem Großteil aus dem Wall Street Journal. Der Ursprung der Sätze in einer Treebank kann eine Rolle spielen, da Parser mit Hilfe von Treebanks trainiert werden. Dazu mehr in Kapitel 2.3.

2.2 Syntaktisches Parsen

Syntaktisches Parsen wird als die "Aufgabe des Erkennens eines Satzes und des Zuweisens einer syntaktischen Struktur" definiert. Ein Parser ist damit ein Programm, das Sätze parst. Zum Einstieg in das Kapitel wird das Parsen als Suche betrachtet. Das Suchproblem besteht darin, aus allen möglichen Bäumen, welche sich mit der Grammatik generieren lassen, den korrekten Baum zur Eingabe zu finden. Der Suchraum wird also von der Grammatik festgelegt. Ein Baum ist korrekt, wenn *S* die Wurzel ist und exakt die Eingabe abgedeckt wird. Anhand dieser zwei Merkmale kann die Suche gestaltet werden. Somit ergeben sich als grundlegende Ansätze die Top-Down und die Bottom-Up Suche.
Beim Top-Down Verfahren wird mit dem Startsymbol *S* begonnen und dieses mit den Regeln der Grammatik Schritt für Schritt erweitert. Bäume deren Blätter nicht auf die Eingabe passen werden abgelehnt.
Die Bottom-Up Suche beginnt, dem Namen entsprechend, am anderen Ende des Baumes. Im ersten Schritt gibt es nur die Eingabeworte als Blätter. Es wird mit den rechten Seiten der Grammatikregeln der Baum nach oben gebaut. Hier kann also ein Baum ausgeschlossen werden, wenn seine obersten Knoten in keiner Kombination auf keiner rechten Seite einer Produktion vorkommen. So werden mit der Top-Down Suche keine Bäume gebaut die niemals das Start Symbol als Wurzel haben, dafür

wird aber die Eingabe im Allgemeinen nicht abgedeckt. Beim Bottom-Up Ansatz verhält es sich genau andersherum.
Das Hauptproblem, welches sich beim Finden des korrekten Baumes ergibt, ist die Mehrdeutigkeit. Zum einen können Wörter mehrdeutig sein, wie etwa das englische Wort *book*, welches sowohl Verb als auch Nomen ist. Zum anderen, und für diesen Kontext relevanter, gibt es die strukturelle Mehrdeutigkeit. Ein Beispielsatz hierfür ist:

> I shot an elephant in my pajamas.

In dieser Satzstruktur kann sich *in my pajamas* sowohl auf den Erzähler, als auch auf den Elefanten beziehen und gibt es mehr als einen korrekten Baum.
Diese Art der strukturellen Mehrdeutigkeit ist die Anhangs-Mehrdeutigkeit. Eine Komponente des Satzes, in diesem Fall die Präpostionalphrase, kann an mehreren Stellen angehängt werden. Kombiniert man die Präpositional- an die Verbalphrase hat der Satz die Bedeutung, dass das Schießen im Pyjama stattgefunden hat. Bindet man diese an das Nomen Elefant wird ausgesagt, dass dieser sich im Pyjama befindet. Grammatikalische Korrektheit ist in beiden Fällen gegeben.
Eine andere Art ist die Koordinations-Mehrdeutigkeit, welche in Verbindung mit Konjunktionen und Satzverbindungen auftritt. Beispielsweise kann der Satzausschnitt *old men and women* unterschiedlich interpretiert werden. *Old* kann sich auf *men and women* oder nur auf *men* beziehen.
Die Anzahl an unterschiedlichen und dennoch grammatikalisch korrekten Bäumen für einen Satz kann also groß sein. Von diesen Bäumen beschreibt aber nur einer den Inhalt des Satzes so, wie er vom Autor gemeint ist. Ein Parser braucht also weitere Kriterien um sich zwischen den verschiedenen Möglichkeiten entscheiden zu können. Hierzu mehr in Kapitel 2.3. Ohne zusätzliche Informationen kann der Parser nur alle möglichen Bäume erstellen. Um das effizient zu bewerkstelligen bietet sich dynamisches Programmieren an.

2.2.1 Dynamische Programmierung

Dynamisches Programmieren ist hier sinnvoll, da beim Erstellen des Baumes im Allgemeinen Mehrfacharbeit anfällt. Baut der Parser zum Beispiel die Bäume von oben nach unten und versucht dabei die Eingabe von links nach rechts abzudecken, dann findet er in der Regel einen Baum der einen Teil des Satzes abdeckt. Da noch nicht die gesamte Eingabe enthalten ist, handelt es sich nicht um einen korrekten Baum. Dennoch kann dieser Baum als Teilbaum in tatsächlichen Lösung enthalten sein. Der Parser müsste ihn aber ohne dynamisches Programmieren immer wieder neu finden. Als Algorithmen zum Parsen haben sich das Chart Parsing, der Earley Algorithmus und der Cocke-Kasami-Younger Algorithmus etabliert.

2.3 Statistisches Parsen

In diesem Kapitel werden Mittel vorgestellt, welche dem Parser helfen sich zwischen mehreren, grammatikalisch korrekten Bäumen eines Eingabesatzes zu entscheiden. Hierfür benötigt er für jeden errechneten Baum zusätzlich die Wahrscheinlichkeit mit welcher dieser semantisch korrekt ist. Das heißt, jede Lösung ist mit einer Wahrscheinlichkeit versehen und es kann diejenige, dessen Wert am höchsten ist als Lösung gewählt werden.

2.3.1 Probabilistische Kontextfreie Grammatiken

Die Anforderung, einem Baum eine Wahrscheinlichkeit zuzuweisen, lässt sich mit dem Konzept der probabilistischen kontextfreien Grammatiken (abgekürzt: PCFG) umsetzen. Abgesehen von zwei Erweiterungen haben die Produktionen einer PCFG die gleiche Form wie die der ursprünglichen kontextfreien Grammatik. Erstens wird jeder Regel eine Wahrscheinlichkeit p, mit $0 \leq p \leq 1$, hinzugefügt.

$$A \rightarrow \beta[p] \tag{2.2}$$

Diese gibt an, mit welcher Wahrscheinlichkeit die rechte Seite und der Vorbedingen der linken Seite auftritt.

$$p := P(A \rightarrow \beta|A) \tag{2.3}$$

Es handelt sich also um eine bedingte Wahrscheinlichkeit.
Zweitens muss für jedes Nichtterminal die Summe der Wahrscheinlichkeiten aller seiner Produktionen 1 ergeben.

$$\sum_{\beta} P(A \rightarrow \beta) = 1 \tag{2.4}$$

Die Wahrscheinlichkeit für einen Baum errechnet sich dann durch das Produkt der Wahrscheinlichkeiten aller verwendeten Regeln. Eine Grammatik heißt konsistent, falls die Summe der Wahrscheinlichkeiten aller möglichen Sätze 1 ergibt. Inkosistenz tritt auf, falls eine Regel der Form $A \rightarrow A$ gibt.
Mit dieser neuen Art von Grammatik ergeben sich auch neue Algorithmen zum Berechnen der Bäume. Sowohl der CKY als auch der Earley Algorithmus sind um den Faktor der Wahrscheinlichkeit erweiterbar, wobei der CKY Algorithmus mehr Verwendung findet.
Einen Nutzen kann man aus der zugefügten Wahrscheinlichkeit nur dann ziehen, wenn ihr numerischer Wert Sinn ergibt. Um diesen Wert für jede Regel zu berechnen gibt es zwei Möglichkeiten. Zum einen kann dieser aus einer vollständig annotierten

Treebank errechnet werden. Hierfür wird jedes Auftreten einer Regel und des entsprechenden Nichtterminals gezählt und dividiert:

$$P(A \to \beta) = \frac{Anzahl(A \to \beta)}{\sum_{\gamma} Anzahl(A \to \gamma)} = \frac{Anzahl(A \to \beta)}{Anzahl(A)} \tag{2.5}$$

Falls man keine solche Treebank zur Verfügung hat, gibt es noch eine zweite Möglichkeit die Werte der PCFG festzulegen. Hierzu arbeitet der Parser einen unannotierter Textkorpus durch und versieht die Sätze mit Tags. Zu Beginn haben alle Produktionen eines Nichtterminals die selbe Wahrscheinlichkeit. Der Korpus wird iterativ durchlaufen und nach jedem Durchgang werden die Wahrscheinlichkeiten der Regeln angepasst. Das Anpassen passiert wie in der ersten vorgestellten Möglichkeit, da zu diesem Zeitpunkt eine annotierte Treebank vorhanden ist. Das Verfahren endet, wenn die Wahrscheinlichkeiten konvergieren.
Die resultierenden Werte in der Grammatik hängen also davon ab, mit welchem Korpus sie berechnet wurden. Hierbei spielen Größe und Textart eine große Rolle. Um beim Parsen eines Textes möglichst gute Ergebnisse zu erhalten, sollte der Parser eine PCFG verwenden, welche mit einem Text des selben Genres erstellt wurde. So kann man beispielsweise mit einer Treebank aus technischen Handbüchern, unabhängig von ihrer Größe, beim Parsen eines privaten Briefes keine guten Ergebnisse erwarten, da sich die Sprache zu sehr unterscheidet.
Außerdem weißt dieses neue Konzept auch zwei Nachteile auf. Das erste Problem der PCFG ergibt sich aus der Kontextfreiheit. Die Wahrscheinlichkeit einer Produktion ist immer gleich, egal an welcher Stelle im Satz sie auftritt. In der tatsächlichen natürlichen Sprache ist das aber im Allgemeinen nicht der Fall. Beispielweise kann die gewählte Produktion einer Nominalphrase abhängig davon sein ob die Phrase Objekt oder Subjekt des Satzes ist. Da diese Information ohne weiteres nicht in der Grammatik nicht berücksichtigt werden kann muss der Mittelwert aus beiden Fällen gebildet werden. Dies führt dazu, dass entweder im Falle des Objekts oder des Subjekts häufig die falsche Regel angewendet wird.
Als zweite Schwachstelle ergibt sich, dass die einzelnen Wörter eine zu kleine Rolle spielen. Als Beispiel hierfür dient die bereits erklärte Anhangs-Mehrdeutigkeit aus Kapitel 2.2. Wiederum wird eine Präpositionalphrase betrachtet, welche entweder an eine Verbal- oder eine Nominalphrase angebunden wird. An welche von beiden hängt wieder allein von der Treebank ab. Dort kommt entweder die Produktion

$$VP \to \alpha \;\; NP \;\; PP$$

oder die Kombination aus

$$VP \to \alpha \;\; NP$$

und

$$NP \to NP \;\; PP$$

öfter vor. α steht für eines der hier möglichen Verb-Nichtterminale, welches aber in beide Fällen immer das selbe ist und deswegen keine Rolle spielt. Wesentlich bessere Resultate sind hier erzielbar, wenn man das Verb aus *VP*, das Nomen aus *NP* und die Präposition aus *PP* in die Entscheidungsfindung mit einbezieht. Es kann aus der Treebank die Information gewonnen werden, ob die gegebene Präposition sich öfter auf das Nomen oder das Verb bezieht. Angenommen es gibt eine Präposition die ausschließlich mit Verben in Verbindung steht. In der Treebank sind es nun aber die Nominalphrasen, an welche öfter Präpositionalphrasen gebunden werden. Dann wird die eben angenommene Präposition mit einer PCFG, welche mit der Treebank errechnet wurde, immer falsch zugeordnet.
Diese Schwäche der PCFG macht sich ebenso bei Koordinations-Mehrdeutigkeit bemerkbar. Beim Verbinden von Phrasen durch Konjunktionen kann wieder die konkrete Konjunktion und die Beziehung zu den entsprechenden Wörtern der zu verbindenden Phrasen betrachtet werden. Betrachtet man nochmal das Beispiel aus 2.2: *old men and women*. Hier könnte eine Treebank die Information liefern, dass die Wörter *men* und *women* per *and* öfter direkt miteinander verbunden werden, als dass nur eines von beiden das Adjektiv *old* zugeordnet bekommt.
Eine mögliche Verbesserung der PCFG ergibt sich durch das Erweitern der Nichtterminale um die Information wessen Kind es ist. Es wird also ein *NP* als *NP^S* geschrieben, wenn *S* der Elternknoten ist oder als *NP^VP*, falls es *VP* ist. Hierdurch ergeben sich zwei neue Regeln in der Grammatik und damit auch zwei neue Wahrscheinlichkeiten. Mit diesem Konzept kann dem Nichtterminal, obwohl die Kontextfreiheit im grammatikalischen Sinne nicht verletzt wird, ein Kontext gegeben werden. Allerdings wird, wenn jedes Nichtterminal für jeden möglichen Elternknoten eine neue Produktion erhält, die Grammatik enorm aufgeblasen. Nebeneffekt dieser neuen Größe ist, dass bei gleich bleibender Treebank für jede Regel weniger Trainingsdaten zur Verfügung stehen und damit die erhaltenen Wahrscheinlichkeitswerte ungenauer sind. Für einen gegebenen Korpus muss also ein Split-and-Merge Algorithmus ausgeführt werden, der errechnet wie weit eine Aufteilung der Nichtterminale sinnvoll ist.

2.3.2 Probabilistische Lexikalisierte Kontextfreie Grammatiken

Ein weiterer Ansatz, um bessere Parserergebnisse zu erhalten, ist das Lexikalisieren der Grammatik. Hierfür muss der Begriff des Kopfes, im Englischen Head, eingeführt werden. Die Idee ist, dass jede syntaktische Einheit einen Kopf, in Form eines Wortes aus dieser Einheit, besitzt. Es wird das Wort genommen, welches "im Satz am grammatikalisch wichtigsten ist". Da jedes Nichtterminal einer Syntaktischen Einheit oder einem POS-Tag entspricht, kann jedem Nichtterminal und jeder Produktion der Grammatik ein Kopf zugewiesen werden. Für das Finden des richtiges Wortes gibt es verschiedene Schritte. Begonnen wird indem jedes POS-Nichtterminal sein

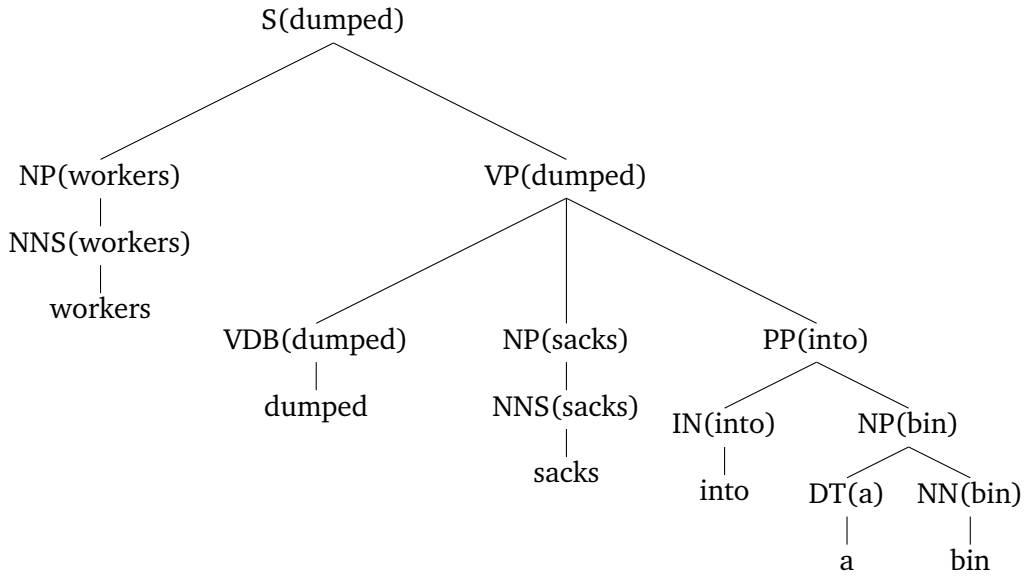

Fig. 2.2: Lexikalisierter Baum, entnommen aus

entsprechendes Kind als Head wählt. Dieses Wort wird dann im Baum nach oben weitergeben. Für jede syntaktische Einheit gibt es Regeln, anhand derer einer der Köpfe der Kinder ausgewählt und als eigener eingesetzt wird. Ein Beispiel ist der lexikalisierte Baum für den Satz /textitworkers dumped sacks into a bin, dargestellt in Abbildung **??**.

Zusätzlich zum Kopfwort kann auch noch dessen POS-Tag abgespeichert werden. Somit wird *S(dumped)* zu *S(dumped, VBD)* und *NNS(workers)* zu *NNS(workers, NNS)*. Die Wahrscheinlichkeit der Regel

$$VP(dumped, VBD) \rightarrow VBD(dumped, VBD)\ \ NP(sacks, NNS)\ \ PP(into, IN) \tag{2.6}$$

ergibt sich wieder aus dem Inhalt der Treebank, nämlich durch die Formel

$$\frac{Anzahl(VP(dumped, VBD) \rightarrow VBD(dumped, VBD)\ NP(sacks, NNS)\ PP(into, IN))}{Anzahl(VP(dumped, VBD))} \tag{2.7}$$

Dieser Wert ist 0, falls der Korpus keinen Satz mit *dumped sacks into* enthält. Existiert kein Satz in welchem sich *VP(dumped, VBD)* finden lässt, so ist dieser Wert nicht definiert.

Aufgrund dieses Problems bedarf es anderer Berechnungsvorschriften, wie zum Beispiel Collins Model 1. Es wird jede Produktion folgendermaßen betrachtet: H ist der Kopf, L_i sind die Nichtterminale links und R_i die rechts davon. Alle Nichtterminale bleiben weiterhin lexikalisiert. Das Kopfwort wird mit h bezeichnet. Damit ergibt sich die Form

$$A \rightarrow L_n ... L_1 H R_1 ... R_m$$

Zusätzlich wird an der Stelle L_{n+1} und R_{m+1} das Nichtterminal *STOP* eingefügt um anzuzeigen, dass hiernach die Regel zu Ende ist. In drei Schritten wird die Wahrscheinlichkeit der Produktion errechnet:

1. Es wird der Kopf mit der Wahrscheinlichkeit $P_H(H|A,h)$ generiert.

2. Alle Elemente rechts vom Kopf werden mit $\prod_{i=1}^{m+1} P_R(R_i|A,h,H)$, also einschließlich dem *STOP*, generiert.

3. Alle Elemente links von *H* werden mit $\prod_{i=1}^{n+1} P_L(L_i|A,h,H)$ generiert.

Für die Regel 2.6 errechnet sich die Wahrscheinlichkeit über

$$\begin{aligned} P = & P_H(VBD|VP, dumped) \\ & \times P_R(NP(sacks, NNS)|VP, dumped, VBD) \\ & \times P_R(PP(into, IN)|VP, dumped, VBD) \\ & \times P_R(STOP|VP, dumped, VBD) \\ & \times P_L(STOP|VP, dumped, VBD) \end{aligned} \tag{2.8}$$

Im Gegensatz zu vorher muss also nicht die Kombination aus *NP(sacks, NNS)*, *PP(into, IN)* und *VBD(dumped, VBD)* vorhanden sein. Es genügt, wenn jedes einzeln, als Kind von *VP(dumped, VBD)* auf der entsprechenden Seite des Heads in der Treebank zu finden ist. Darüber hinaus wird das Modell noch um die Information der Distanz erweitert. Es wird also zusätzlich berücksichtigt wie weit ein Nichtterminal vom Kopf der Regel entfernt ist. Der Vollständigkeit halber ist außerdem zu erwähnen, dass es neben Model 1 noch zwei weitere Modelle gibt. Diese Modelle bilden die Grundlage für den Collins Parser.

3 Konpzept

Abbildung **??** zeigt das Konzept nach dem implementiert wurde. Hier wird dargestellt, die Leistung eines Parsers evaluiert wird. Der Input-Text ist der rohe Text den der Parser mit syntaktischer Annotation versehen soll. Das Parser-Modell ist die Grundlage auf der ein Parser arbeitet. Der Goldstandard ist der Input-Text mit, als korrekt angesehener, Annotation. Dieser wird mit der Ausgabe des Parser verglichen. Daraus lassen sich Kennzahlen zur Leistung des Parsers errechnen, die am Ende ausgegeben werden. Um mehrere Parser miteinander zu vergleichen werden die Resultate der einzelnen in eine Tabelle zusammengefasst und dargestellt. Alle Elemente werden nachfolgend ausführlich vorgestellt.

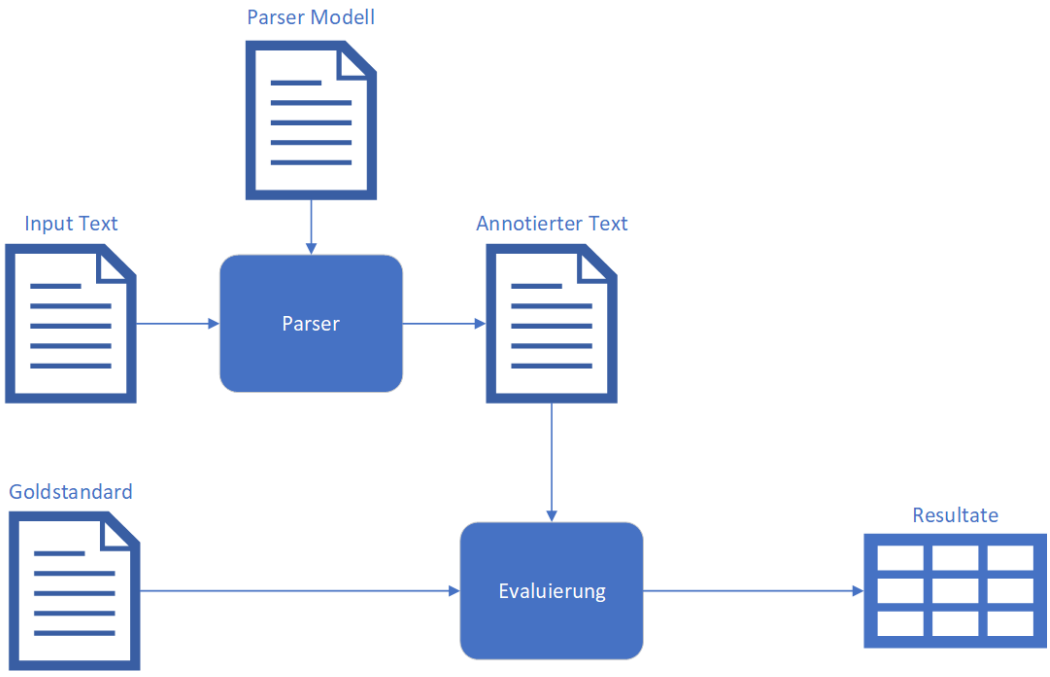

Fig. 3.1: Konzeptueller Aufbau

3.1 Input-Text

Die Eingabe des Parsers sind Sätze, die nach Tokens getrennt sind. Das heißt, alles was ein POS-Tag bekommt wird mit Leerzeichen getrennt. Ein solcher Satz wird

sequenziert genannt. Die meisten Wörter brauchen keine weitere Verarbeitung, da sie bereits ein Leerzeichen zum nächsten Wort haben. Ein typisches Beispiel für die Notwendigkeit der Aufteilung ist *he's*. Hier erkennt der Parser nur dass es sich um die Wörter *he* und *is* handelt, wenn ein Leerzeichen vor dem Apostroph eingeschoben wird. Bekommt man aus einem Korpus keine Version des Satzes, bei dem dieser Arbeitsschritt schon erledigt ist, muss man einen Tokenizer vorschalten. Hierfür gibt es unterschiedliche Anbieter, wie z.B. Stanford oder OpenNLP. Allerdings wird in der Implementierung kein Tokenizer verwendet. Die Sätze der Eingabe müssen über ein Textdokument übergeben und zeilenweise getrennt sein. Der Parser erstellt für jede Zeile einen Baum.

3.2 Parser Output

Als Ausgabe gibt der Parser die annotierten Sätze in der Reihenfolge, in der sie eingegeben wurden, zurück. Es wird hierfür wieder ein Textdokument erstellt. Die annotierten Version des Satzes

> It goes 150 miles an hour .

lautet

> ((S (NP (PRP It)) (VP (VBZ goes) (NP (NP (CD 150) (NNS miles))
> (NP (DT an) (NN hour)))) (. .)))

Das äußerste Klammerpaar hat kein führendes Nichtterminal und könnte deshalb weggelassen werden. Es enthält typischerweise das Startsymbol der Parser, also *ROOT*, *TOP* oder ähnliches. Dieses Nichtterminal wird aber für die Evaluierung aus dem Ergebnisbaum herausgenommen, weil es keine syntaktische Information liefert. Da der Satz des Goldstandards ebenfalls diese unannotierten Klammern besitzt, werden sie aus beiden Bäumen nicht gelöscht sondern im Evaluationsschritt ignoriert.

3.3 Parser Modell

Ein Parser erzielt, je nachdem welches Modell ihm zu Grunde liegt, sehr unterschiedliche Ergebnisse. Daher ist dieses Modell nicht fest im Parser verankert, sondern wird ihm als Datei übergeben. Somit kann man selbst Modelle anhand einer Treebank erstellen lassen um dann beispielsweise auf einem Parser die unter-

schiedlichen Modelle vergleichen. Alle verwendeten Parser bringen ein trainiertes Modell mit.

3.4 Goldstandard

Als Goldstandard werden die korrekten Bäume bezeichnet. Diese wurden entweder von Menschenhand erstellt oder durch ein Parser erstellt und dann von Menschen nachkorrigiert. Es muss für ein eingegebenes Textdokument, welches vom Parser bearbeitet werden soll auch ein Textdokument geben, das für jeden dieser Sätze die annotierte Version enthält. Die gratis verfügbaren Treebanks weisen oft ein unterschiedliches Sortiment an Dateien und Formaten auf.
Der, in dieser Arbeit verwendete, Korpus heißt "The NAIST-NTT TED Talk Treebank" und stammt aus dem Jahr 2014. Er umfasst etwa 23.000 Wörter in 1.200 Sätzen, welche aus 10 Reden gewonnen wurden. Zur Annotierung wurde der Berkeley Parser verwendet und dessen Resultat per Hand verbessert. Als Tag-Set wurde wieder die Vorlage der Penn Treebank verwendet. In dieser Treebank sind für jeden Satz eine Rohversion, eine sequenzierte und eine annotierte Version enthalten. Da es sich bei den Texten um Gesprochenes handelt, sind noch weitere Informationen, wie zum Beispiel Zeitmessung eines Satzes und ähnliches vorhanden. Das wird hier nicht weiter verwendet.
Bei Verwendung anderer Goldstandards können diverse Probleme auftreten. Zwei davon sind hier kurz durchdacht aber nicht implementiert worden.
Falls das Problem auftritt, dass nur die korrekten Lösungsbäume zur Eingabe verfügbar sind, so kann man sich z.B. mit NLTK die Blätter jedes Lösungsbaums ausgeben lassen. Hierdurch erhält man, den in seine Token aufgeteilten Satz.
Ist der Lösungsbaum nicht einzeilig, sondern erstreckt sich ähnlich zu 2.1 über mehrere Zeilen, so muss hier auch eine Vorverarbeitung geschehen. Entweder kann anhand der Klammerung erkannt werden wo die Grenzen des Baumes liegen oder es müssen alle Zeilen, die mit einer Art Leerzeichen (Tabulator, u.ä.) beginnen, an die vorherige gehangen werden.

3.5 Parser

Alle Parser erhalten die Eingabe in selber Form, allerdings kann jeder Parser noch eine individuelle Vorverarbeitung benötigen. Dazu mehr in Abschnitt ... . Im Rahmen dieser Arbeit sind drei Parser verglichen worden. Das sind der Stanford-Parser, der Berkeley-Parser und der Parser aus dem OpenNLP Paket. Jeder verwendet zum Annotieren die Penn Treebank Tags und liefert die Ausgabe standardmäßig ohne die zusätzlichen relationalen Tags.

3.5.1 Stanford-Parser

Der Stanford-Parser bringt ein Modell namens *englishPCFG* mit. Es handelt sich dabei um eine unlexikalisierte PCFG für die englische Sprache.

3.5.2 Berkeley-Parser

3.5.3 OpenNLP-Parser

3.6 Evaluierung

Für die Evaluierung wird das Ergebnis der Parser zeilenweise mit dem Goldstandard verglichen. Zu diesem Zweck werden die Konstituenten der Bäume betrachtet. Eine Konstituente beschreibt einen Knoten im Baum und enthält die Information über das Nichtterminal, den Start- und den Endpunkt. Die Punkte geben den Index des ersten und letzten Wortes dieses Teilbaums an. Mit jedem Wort wird der Zähler inkrementiert. Für den Beispielsatz *They learn much faster.* (Baum in Abbildung 3.2) sind die Konstituenten in Tabelle 3.1 dargestellt.

Falls es, für eine Konstituente des Parsers, im Goldstandard eine mit gleichem

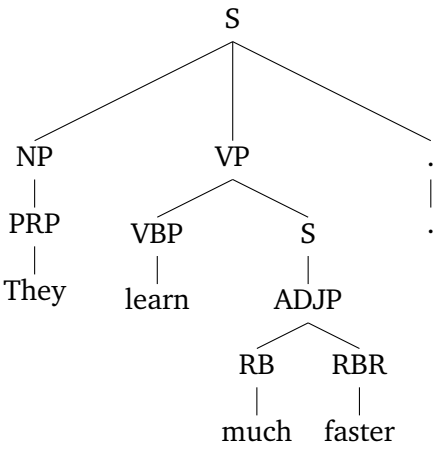

Fig. 3.2: Syntaxbaum zu *They learn much faster.*

Nichtterminal und gleicher Spanne gibt, wird diese als richtig bewertet. Hier muss bei der Ausführung entschieden werden, welche Nichtterminale gewertet werden. Zum einen kann man alle in das Resultat einbeziehen. Zum anderen können die POS-Tags herausgelassen und nur die syntaktischen Tags aus Tabelle 2.2 bewertet werden. Der Grund ist, dass man so Parser einbringen kann, die als Eingabe einen mit POS-Tags versehenen Text bekommen. Diese Tags verfälschen dann die Korrektheitsrate der Parser und müssen ignoriert werden. Da in dieser Arbeit ausschließlich Parser genutzt werden, die als Eingabe sequenzierten Text bekommen, betrachten

Nichtterminal	Text	Start	Ende
S	They learn much faster .	0	5
NP	They	0	1
PRP	They	0	1
VP	learn much faster	1	4
VBD	learn	1	2
S	much faster	2	4
ADJP	much faster	2	4
RB	much	2	3
RBR	faster	3	4
.	.	4	5

Tab. 3.1: Konstituenten zu Abbildung 3.2

wir hier den Fall in dem alle Tags relevant sind. Der Evaluierungsmechanismus ist angelehnt an ... Als Bewertungskriterien werden folgende vier Werte errechnet: *Precision*, *Recall*, F_1 und *Relative Cross Brackets* (kurz: RCB). Siehe Formeln 3.1 bis 3.4.

$$Precision = \frac{\#\ korrekte\ \ Konstituenten}{\#\ \ Konstituenten\ \ im\ \ Parseroutput} \tag{3.1}$$

$$Recall = \frac{\#\ korrekte\ \ Konstituenten}{\#\ \ Konstituenten\ \ im\ \ Goldstandard} \tag{3.2}$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.3}$$

$$RCB = \frac{\#\ Parser\ Konst.\ die\ Goldstandard\ Konst.\ kreuzen}{\#\ \ Konst.\ \ im\ \ Parseroutput} \tag{3.4}$$

Die *Precision*, im Deutschen *Genauigkeit*, gibt an, wieviele der Konstituenten des Parsers korrekt sind. *Recall*, oder *Trefferquote*, beschreibt zu welchem Teil die Konsituenten des Goldstandards mit den Parser-Konstituenten abgedeckt wurden. F_α ist das gewichtete harmonische Mittel aus beiden. Im Rahmen dieser Arbeit wurde nur F_1 betrachtet, d.h. beide werden gleich gewichtet. Beim *RCB*-Wert handelt es sich um einen Indikator der ausschließlich für Syntaxbäume eingesetzt wird. Er gibt an wieviele der Konstituenten des Parsers sich mit denen des Standards kreuzen. Kreuzen ist in diesem Kontext folgendermaßen definiert. Seien *k1* und *k2* zwei Konstituenten mit unterschiedlichen Start- und Endpunkten. Außerdem gilt o.B.d.A., dass *k1* den niedrigeren Startpunkt hat. Dann kreuzen sie, falls der Endpunkt von *k1* größer als der Startpunkt und kleiner als der Endpunkt von *k2* ist. Anders ausgedrückt, sie sind nicht ineinander enthalten aber teilen sich mindestens

ein Wort. Aus der Definition ergibt sich, für *k1* aus dem Parser und *k2* aus dem Goldstandard gilt, dass *k1* nicht korrekt sein kann. Angenommen es wäre korrekt, so müsste es ein *k1'* im Goldstandard mit identischen Grenzen geben. *k1'* beginnt vor *k2* und hört auch vor diesem auf und kann somit kein Eltern- oder Kindknoten von diesem sein. Da sie sich aber mindestens ein Wort teilen würde für dieses Wort gelten, dass es Blatt von zwei verschiedenen Teilbäumen wäre, also zwei Eltern hätte. Hier ergibt sich also ein Widerspruch. Aus den Formeln folgt damit außerdem: $RCB \leq 1 - Precision$. Für alle vier Kennzahlen liegt der Wertebereich zwischen 0 und 1. *Precision*, *Recall* und F_1 sollten möglichst hoch sein und *Relative Cross Brackets* möglichst niedrig.

4 Implementierung

4.1 Zielsetzung

List of Figures

List of Tables

Declaration

You can put your declaration here, to declare that you have completed your work solely and only with the help of the references you mentioned.

Bayreuth, Juni 1, 2019

Klaus Freiberger