Grundlagen zu Natural Language Processing und Analyse von GATE als NLP-Tool

Projektarbeit, Bearbeitungszeitraum Sommersemester 2018
Fachhochschule Dortmund
Fachbereich 4, Informatik

Aaron Schul, Felix Ritter

Contents

List of Tables

1 Einleitung

Mit voranschreitender Globalisierung und immer größeren Mengen an übertragenen Informationen steigt auch die Menge an zu verarbeitender natürlicher Sprache. Während dies bisher manuell durch den Menschen geschah, kommt allmählich aufgrund der schieren Menge von Daten vor allem im unternehmerischen Kontext und im Internet die Notwendigkeit auf, die natürliche Sprache automatisiert durch Computer verarbeiten zu lassen.

Dies bezeichnet man als Natural Language Processing (kurz NLP). Es hilft dabei, Texte beispielsweise nach Schlagworten zu durchsuchen, strukturell zu analysieren, Muster zu erkennen und teilweise sogar die Bedeutung von Geschriebenem zu verstehen und diese Semantiken darzustellen.

Mit dem Aufkommen von NLP steigt auch das Interesse an NLP-Tools, also an Programmen, welche in der Lage sind, die für NLP notwendigen Funktionalitäten übersichtlich und praktikabel bereitzustellen. Eines der bekanntesten NLP-Tools ist das von der University Of Stanford entwickelte GATE, welches eine Vielzahl von Anwendungsbereichen im NLP abdeckt.

Das Ziel dieser Arbeit ist es, eine Anforderungsanalyse für solche Tools durchzuführen und diese dann mit dem GATE-Tool abzugleichen. Somit bietet sich die Möglichkeit GATE zu analysieren und daraufhin kriterienbasiert zu evaluieren. Funktionale wie nicht-funktionale Anforderungen werden definiert.

Zu diesem Zweck werden zunächst ausführlich die Grundlagen zu NLP und dessen Tools erläutert. Die Teilschritte der Textverarbeitung werden theoretisch wie praktisch anhand von Implementierungsbeispielen erläutert. Daraufhin können mit Hilfe von Use Cases Anforderungen an diese Tools erhoben werden, auf denen Analyse und Evaluation des GATE-Tools beruhen sollen. Abgeschlossen wird die Arbeit von einer kurzen Zusammenfassung der gewonnenen Erkenntnisse, gefolgt von einem Ausblick auf mögliche zukünftige Arbeiten und Entwicklungen für NLP-relevante Domänen.

2 Grundlagen von Natural Language Processing

2.1 Einführung

3 Einführung

Der folgende Abschnitt befasst sich mit den Grundlagen zu Natural Language Processing (im Folgenden kurz NLP). Dazu gehört neben dessen Herkunft bzw. Motivation eine inhaltliche Wissensgrundlage, welche für den weiteren Verlauf der Ausarbeitung relevant ist.

3.0.1 Motivation und Herkunft

3.1 Motivation und Herkunft

Ein Großteil der menschlichen Kommunikation findet durch natürliche Sprache statt. QUELLE Dies ist bereits seit Jahrtausenden so, jedoch hat sich die Übertragung dieser Sprache mit der Zeit verändert und weiterentwickelt.

Damals lediglich von Mund zu Mund übertragen war der erste große Schritt die Entwicklung der Schrift. Mithilfe von Hieroglyphen, Alphabeten oder anderen Zeichen war man in nun in der Lage, natürliche Sprache über lange Zeit und/oder weite Strecken zu vermitteln. Dies erwies sich als sehr Vorteilhaft und so setzte sich die Entwicklung fort, bis man über das Morsen und das Telefon schließlich die elektronische Nachricht erfand. Im Rahmen der voranschreitenden Digitalisierung und Globalsierung steigt die Menge an übertragener natürlicher Sprache nach wie vor rasant an, sodass heutzutage ein Leben ohne beispielsweise die E-Mail unvorstellbar scheint (OBJEKTIVER: sodass heute etwa E-Mails einen wesentlichen Bestandteil der Kommunikation ausmachen.) . Obgleich die Übertragung durch kabelgebundene oder drahtlose Kommunikation weitgehend automatisiert ist, erfolgt die Auswertung des Inhalts größtenteils manuell durch menschliche Empfänger.

Die Daten selbst sind jedoch inzwischen kaum noch nur durch den Menschen effizient zu verarbeiten, sodass man nach einer neuen Möglichkeit sucht, dem Menschen diese Arbeit zu erleichtern, oder sogar abzunehmen. Mit diese Aufgabe beschäftigt sich NLP.

3.2 Hypothese

3.3 Methodik

3.4 Aufbau der Arbeit

4 Natural Language Processing

4.1 Grundlagen und thematische Einordnung

Der folgende Abschnitt befasst sich mit den Grundlagen zu Natural Language Processing (im Folgenden kurz NLP). Dazu gehört neben dessen Herkunft bzw. Motivation eine inhaltliche Wissensgrundlage, welche für den weiteren Verlauf der Ausarbeitung relevant ist.

4.1.1 Definition

NLP ist grob definiert als die automatische oder halb-automatische Verarbeitung von natürlicher Sprache. (QUELLE) Manche schließen aus dieser Definition die Erfassung der Sprache aus, da diese nicht Teil der eigentlichen Verarbeitung ist. (Der Einfachheit halber wird im weiteren Verlauf davon ausgegangen, dass die Sprache als digitale Textdatei vorliegt, wenn nicht explizit anders angegeben. Es kann sich jedoch auch um gesprochenes Wort oder Gesten handeln)

NLP überschneitet sich mit vielen wichtigen Bereichen der Wissenschaft. Hauptsächlich sind dies Linguistik und Informatik, allerdings fließen auch Psychologie, Philosopie und Mathematik bzw. Logik stark mit ein.

4.1.2 Ziel und funktionale Einordnung

Das Ziel von NLP ist die Extraktion von Informationen aus gegebenen Texten, also aus einem natürlichsprachigen Input einen Wissensoutput über den Inhalt des Dokuments zu generieren. Dies wird mithilfe von NLP-Tools wie GATE umgesetzt.

Dazu ist, wie im nächsten Abschnitt beschrieben, die linguistische Analyse der Sprachstruktur der Dokumente in Teilschritten erforderlich. Auf Basis der syntaktischen Analyse kann dann eine semantische Analyse durch Einsatz sogenannter Ontologien (Sammlung von Wissen aus einer Domäne)

erfolgen. Letztendlich kann somit tatsächlich Wissen über die Bedeutung aus dem Dokumenteninhalt gewonnen werden.

4.1.3 Beispiele früher Entwicklungen

Die Forschungsergebnisse des Linguisten Noam Chomsky seit den 1950er Jahren haben aufgezeigt, dass es grundsätzlich möglich ist, einge Aspekte der Regeln englischer Sprache zu formalisieren. In Kombination von beispielsweise Automaten und generativen Grammatiken (Abschnitt 2.4 und 2.6), die auch *Chomsky-Hierarchie* genannt wird, wird verdeutlicht, wie diese automatisiert aufgefasst und nachvollzogen werden kann. Siehe dazu auch [CHO57]. Chomsky schaffte damit zum großen Teil die Grundlagen der Modelle heutiger Sprachanalyse-Tools.

Bereits in den 1960er Jahren versuchte man durch Sprachcomputer bzw. Chatbots wie *ELIZA* (Weizenbaum 1966) die Mensch-Maschine-Kommunikation umzusetzen, indem die Textnachricht eines Benutzers automatisch durch einen Computer verarbeitet und eine plausible Antwort gegeben wurde, damals jedoch noch ohne echte Wissensbasis. Aus heutiger Sicht kann man ELIZA somit keine Möglichkeit zur "intelligenten" Kommunikation attestieren, da scheinbar der Sinn hinter den Nutzereingaben nicht gänzlich erfasst wird.

Heutige Entwicklungen im Bereich Sprachassistenzsysteme sind alltäglicher Begleiter jedes Smartphone-Nutzers geworden; sie analysieren die Spracheingaben mithilfe von NLP-Techniken bezüglich ihrer Bedeutung in Echtzeit über Clouds. (QUELLE)

4.1.4 NLP als Annäherung an natürlichsprachliche Probleme

Bereits bei der Mensch-Mensch-Kommunikation sind Verständigungsprobleme vorhandenen und treten unwillkürlich auf. Die Intention einer Aussage ist etwa nicht nur abhängig von dem, was tatsächlich gesagt wird, sondern auch etwa von Gestik, Mimik und der Situation, in der kommuniziert wird. Bei der Untersuchung von geschriebenem Text in natürlicher Sprache treten diese Betrachtungen jedoch in den Hintergrund, da der Inhalt im Vordergrund steht und mitunter die Situation des Autoren unklar oder irrelevant ist. Wissen aus der Domäne ist jedoch zwingend für das Verständnis spezifischer Fachbegriffe von Nöten, daher müssen auch automatische NLP-Tools diese einbeziehen können. Zudem gibt es oft Veränderungen der Bedeutung von Sprache die stark kontextabhängig sind, wie zum Beispiel Sarkasmus

oder Ironie. Auch verändern Worte im Laufe der Zeit ihre Bedeutung oder durch Rechtschreibreformen ihr Aussehen. Hier soll nicht weiter auf Aspekte der Kommunikationswissenschaften eingegangen werden, jedoch ist beispielsweise die Ambiguität einer Aussage ganz alltäglich und hat gleichsam verschiedene Auswirkungen. Solche Mehrdeutigkeiten werden durch syntaktische und semantische Fehlinterpretationen verursacht. (QUELLE)

Wo sich Menschen dabei im Zweifel auf Erfahrungen und spezifische Fachkenntnisse verlassen oder bei ihrem Kommunikationspartner nachfragen können, müssen sich Computer allein auf die vorliegenden Dokumente in Schriftform verlassen; sie wissen nichts über den Kontext des Dokuments. Wie später gezeigt wird, kann jedoch etwa Fachwissen durch domänenspezifische Wissensbasen (ABSCHNITT?) simuliert werden.

Die Frage nach einer intelligenten, algorithmen- und / oder wissensbasierten Auswertung durch Computer stellt sich im Angesicht der aufgezeigten Besonderheiten natürlicher Sprache. Es gilt also, für NLP nicht nur die Probleme zu lösen, die generell mit präziser Automatisierung verbunden sind, sondern zusätzlich so zuverlässig wie möglich die oben genannten und der Sprache inhärenten Komplikationen zu bewältigen. Bereits einfache Anfragen können computergestützte Frage-Antwort-Systeme wie SQL-Datenbanken an ihre Grenzen bringen, wie die verschiedenen Formen von Mehrdeutigkeit zeigen können:

Schon das Wort *überblicken* kann das Übersehen von etwas, ebenso wie das im Blick haben (Gegenteil) bedeuten.

Der Satz *"Die Betrachtung des Studenten"* kann etwa als Student, der (etwas) betrachtet, oder als ein Student, der (von jemand anderem) betrachtet wird, verstanden werden. Sätze mit solch einem Satzbau (bestehend aus nominalisiertem Verb, Bezugswort und Substantiv) verursachen unwillkürlich zwei mögliche Deutungen aufgrund der unklaren Syntax.

Die Frage *"Verkaufen Sie Handys und Computer von Samsung?"* kann als Frage nach allgemeinen Handys und nur Computern, spezifisch von Samsung oder nach Handys und Computern gleichermaßen nur von Samsung verstanden werden. Auch hier entsteht Unklarheit durch zwei mögliche Bezüge des einschränkenden Relativsatzes. Ob eine Aufzählung oder ein Unterschied (zwischen *von Samsung* und *nicht von Samsung* gemacht wird, ist syntaktisch unklar.)

"Wie schnell ist der Bus?" und *"Wie schnell ist der Bus da?"* mögen zunächst ähnlich aussehen, fragen jedoch unterschiedlichen Inhalt ab und haben nichts miteinander zu tun. Außerdem könnte der zweite Satz kon-

textabhängig etwa nach der Fahrzeit bis zum Ziel oder nach der Ankunftszeit, an der man einsteigen kann, fragen. Die vage Formulierung von Sätzen stellt bei der Frage nach deren Bedeutung eine Herausforderung dar, durch Untersuchung des Kontextes der Frage kann hier jedoch meist recht treffsicher geantwortet werden.

Abgesehen von diesen syntaktischen Uneindeutigkeiten muss NLP auch Hürden der inhaltlichen Mehrdeutigkeit bzw. Semantik bewältigen. Aktuell beschäftigt man sich zum Beispiel mit der Aufgabe zu erkennen, ob ein Kommentar zu einem Video der Webseite *Youtube* eher positiv oder eher negativ gesinnt ist. Was hier für den Menschen schon beim ersten Lesen sofort erkennbar wird, stellt für den Computer ein ernstzunehmendes Problem dar. Tonalität und Stimmung des Kommentars, der unter einem Video als Reaktion entsteht, sind mitunter nicht direkt aus dem audiovisuellen Eindruck aus dem Video ersichtlich. (?) Solche scheinabr simplen Probleme sind für den Menschen recht einfach zu lösen. Wie oben erwähnt existiert NLP bereits seit ca. Mitte des 20. Jahrhunderts, jedoch ist die Behandlung natürlichsprachlicher Probleme auch heute noch problematisch in der Implementierung. Wenn Systeme die oben genannte Beispiele nicht explizit ausschließen, muss die Verarbeitungslogik besondere Rücksicht auf Uneindeutigkeiten nehmen.

4.2 Konzepte aus der Logik für NLP-Anwendungen

Hier vllt was zu εT Logik

4.3 Linguistische Analyse

Der folgende Abschnitt beschäftigt sich mit der Analyse der sprachlichen Struktur eines Dokumentes. Am Anfang von NLP steht die linguistische Analyse des Textes etwa bezüglich Satzstruktur und Differenzierung von Wörtern aus der Sprache und Eigennamen. Das Ziel ist die sogenannte *Annotation* der Textbausteine bezüglich ihrer Bedeutung. Sachzusammenhänge und Relationen von Wörtern sollen dargestellt werden. (?) Der folgende Abschnitt beschäftigt sich mit der Analyse der sprachlichen Struktur eines Dokumentes. Am Anfang von NLP steht die linguistische Analyse des Textes etwa bezüglich Satzstruktur und Differenzierung von Wörtern aus der Sprache und Eigennamen. Das Ziel ist die sogenannte *Annotation*

der Textbausteine bezüglich ihrer Bedeutung. Übergreifende Sachzusammenhänge anhand der Relationen von Wörtern sollen dargestellt werden. Textstellen, die thematisch zueinander passen, sind im Ergebnis der Analyse so annotiert. (?)

NLP kann, wie später in dieser Arbeit beschrieben, als Zusammenfassung von 6 wesentlichen Teilbereichen verstanden werden. Begonnen bei der einfachen Wortanalyse, werden später Bedeutungen und Verknüpfungen der Aussage aus Sätzen oder Wörtern auf die reale Welt analysiert. Die Linguistik befasst sich seit jeher mit der Untersuchung von natürlicher Sprache im Hinblick auf die dahinter liegenden Konstrukte. Die folgenden Teilbereiche, die daraus bekannt sind, sind bei der natürlichsprachigen Analyse identifiziert worden: NLP kann, wie später in dieser Arbeit beschrieben, als Umsetzung vierer wesentlicher Teilbereichen aus der Linguistik verstanden werden. Begonnen bei der einfachen Wortanalyse, werden später Bedeutungen und Verknüpfungen der Aussage aus Sätzen oder Wörtern auf die reale Welt analysiert. Die Linguistik befasst sich seit jeher mit der Untersuchung von natürlicher Sprache im Hinblick auf die dahinter liegenden Konstrukte. Die folgenden Teilbereiche, die daraus bekannt sind, sind bei der natürlichsprachigen Analyse identifiziert worden:

1. Morphologische Analyse - Die Zerlegung von Wörtern bezüglich ihrer Struktur. Die Komposition aus Präfix, Wortstamm und Suffix von Wörtern wird erkannt, um Fall, Tempus, Numerus etc. des Wortes zu bestimmen. Im Englischen zeigt so die Endung -ed bei Verben die Vergangenheitsform an. Dabei ist zu beachten, dass Mehrdeutigkeiten entstehen können.

2. Syntax - Aufbau von Sätzen durch einzelne Wörter. Jede Sprache besitzt syntaktische Regelungen bezüglich der auftauchenden Wörter; im Deutschen folgt etwa auf einen Artikel irgendwann ein Substantiv oder bestimmte Signalwörter geben Satztyp und Tempus an. Auf Basis dieses Wissens können formale und strukturelle Analysen der Textbestandteile erfolgen.

3. Semantiken - Die Identifikation der Bedeutung von einzelnen Sätzen und Wörtern. Die Semantik eines Textabschnittes wird häufig als "Logik" bezeichnet und fragt nach dessen Bedeutung bzw. Thema. Semantische Deutungen können anhand von Kompositionen einzelner Wörter und Sätzen auf Basis der semantischen Analyse erkannt werden.

4. Kontext - Darstellung von Sachzusammenhängen aus der Vereinigung von ähnlichen Bedeutungen. Textbausteine, die sich mit dem gleichen Thema beschäftigen, werden dem gleichen Sachzusammenhang zugeordnet und als Teil dessen annotiert.

NLP spaltet bei der Analyse diese Sprachbestandteile aus der Linguistik weiter in Teilbereiche auf. NLP spaltet bei der Analyse diese Sprachbestandteile aus der Linguistik weiter in Teilbereiche auf und ordnet diese Aufgaben in der Implementierung Programmbausteinen zu.

4.3.1 Allgemeine NLP Architektur

Die Verarbeitung eines Dokumentes mittels NLP erfolgt nacheinander in einzelnen Schritten, die jeweils einen Bereich oder Teilbereich der Linguistik des Textes abbilden. Die Analyse erfolgt dabei nacheinander und wird immer feingranularer, begonnen bei einfachen grammatikalischen Analysen bis hin zu komplexen Wissensbasierten verfahren. Die folgenden Schritte stellen laut [COP04] eine typische komponentenweise Architektur dar. Es ist anzumerken, dass auch mehrere Schritte gemeinsam in einem Schritt durchgeführt oder gar ganz entfallen können, abhängig von der Umsetzung anderer Stufen:

1. Input Processing - Erkennung der Dokumentensprache und Normalisierung des Textes. Im ersten Schritt geht es um das korrekte Format des Eingabedokumentes für die Verarbeitung. Dieser Vorgang stellt jedoch noch keine Analyse dar, sondern dient lediglich der Vorbereitung.

2. Morphologische Analyse - Die meisten Sprachen sind im Bezug auf ihre Grammatik und syntaktische Struktur der Wörter in Systemen abgeschlossen, etwa durch Modellierung mittels Automaten zur Erzeugung der Worte. Auch können Vokabeln in Wörterbüchern/Lexika gesammelt und Regeln auf ihnen formuliert und gespeichert werden.

3. Part-of-Speech-Tagging - Die einzelnen Wörter eines Satzes werden im Hinblick auf den Sach- oder Satzzusammenhang, wie auch auf die Stellung der Stellung im Satz hin analysiert. Basierend auf Kontext und/oder Erfahrungswerten bzw. Heuristiken können die Wörter korrekt erfasst und deren Fall getaggt werden. Subjekte, Prädikate usw. werden identifiziert. Dazu ist eine Wissensbasis mit Trainingsdaten und die Einordnung des Kontextes darin von Nöten.

4. Parsing - Die Ergebnisse der der vorherigen Schritte werden weiter verarbeitet und in ein standardisiertes Format gebracht. Syntaktische Zusammenhänge, wie etwa das Zusammenfassen von Wörtern zu einer gemeinsamen Bedeutungsphrase und das Zuordnen von Verben zu einem Nomen können nach dem Parsing dargestellt werden.

5. Disambiguation - Die Entfernung von Mehrdeutigkeiten auf Basis der Ergebnisse des Parsings stellt einen entscheidenden Schritt für die semantischen Analysen dar. Nur wenn die Aussage und der Kontext eines Satzes klar erkennbar sind, kann dessen Semantik gezielt abgeleitet werden.

6. Context Module - Textbausteine, deren Interpretation semantisch auf dem Kontext anderer Sätze oder Wörtern beruhen, können erfasst werden. Bei Anaphern ist etwa das Verständnis des aktuellen Kontext abhängig von einer vorherigen Deutung.

7. Text Planning - Die Sachzusammenhänge und Semantiken, die aus dem zugrundeliegenden Text extrahiert wurden, werden für die Darstellung im fertigen Text definiert. Es wird festgelegt, welche der Bedeutungen qualitativ übertragen und vermittelt werden sollen. Die Reihenfolge und Umfang der behandelten Themen wird geschätzt und definiert.

8. Tactical Generation - Die Bedeutungen werden in Form konkreter Zeichenketten generiert, die die gewünschte Bedeutung enthalten. Diese Textbausteine basieren häufig direkt auf den Ergebnissen des vorherigen Parsings, da dort schon Bedeutungen zusammengefasst werden können. Der quantitative Teil des Textes wird erzeugt.

9. Morpholigical Generation - Die erzeugten Sätze werden morphologisch an die Rahmenbedingungen der verwendeten Sprache angepasst. Grammatiken und Regeln der Satzkonstruktion werden auf die erzeugten Wörter angewendet, sodass ein lesbarer und nachvollziehbarer Text entsteht.

10. Output Processing - Der Text wird nun, nachdem dieser inhaltlich nun komplett erzeugt vorliegt, an das Design und Format des jeweiligen Einsatzzwecks angepasst. Sollte Text, anders als in dieser Arbeit der Einfachheit halber angenommen, nicht als geschriebenes Wort ohne besondere Formatierung vorliegen, kann dieser einem Formatter zur

Designanpassung übergeben werden. Ferner kann etwa mihilfe von Text-to-Speech eine Audioausgabe erfolgen.

4.4 Morpholoie in NLP

Der folgende Abschnitt befasst sich genauer mit der morphologischen Analyse von Wörtern und den dazu verwendeten Zustandswandlern. Da in der englischen Sprachanalyse bereits die meisten Fortschritte erzielt wurden und sie verhältnismäßig wenig komplex im Vergleich zu anderen Sprachen ist, werden im Folgenden viele der Beispiele auf Englisch behandelt. (? vielleicht trennen)

Wie bereits erwähnt, befasst sich die Morphologie mit der Struktur, Zusammensetzung und Korrektheit einzelner Wörter. Jedes Wort der englischen Sprache besteht aus mindestens einem Wortstamm und beliebig vielen Affixen. Affixe sind unterteilt in Präfixe und Suffixe, abhängig davon, ob sie vor oder hinter dem Wortstamm auftauchen. Es gibt auch Affixe, welche in der Mitte des Wortes eingeordnet werden. Diese existieren jedoch nicht in der englischen Sprache und sind daher künftig zu vernachlässigen. (? Reihenfolge)

Man betrachte beispielsweise das Wort *"believe"* von dem englischen Verb *"to believe"* (zu Deutsch: "glauben"). (kürzer) Hängt man das Suffix *"-able"* an, so erhält man das Adjektiv *"believable" ("glaubhaft").* Fügt man nun das Präfix *"un-"* hinzu erhält man die Negation *"unbelievable" ("unglaubhaft").* Die morphologische Analyse erkennt dies nicht nur als ein Wort, sondern als Präfix-Stamm-Suffix-Konstrukt.

Es wird zwischen ableitender und flexionaler Morphologie unterschieden, wobei die Abgrenzung jedoch nicht ganz eindeutig ist. Affixe wie *"anti-"*, *"re-"* und das oben verwendete *"un-"* gelten als ableitend und können mehrfach und sogar rekursiv zu Stämmen auftauchen. Sie fügen dem Stamm in der Regel eine Zusatzinformation hinzu, wie z.B. Negation. Es kann auch, wie in dem Beispiel oben demonstriert, zur Änderung der Wortart kommen. Flexionale Affixe werden normalerweise aufgrund grammatikalischer Regeln wie Pluralisierung hinzugefügt. Hierzu gehören das Plural-"-s" und die Endung *"-ed"* in der Vergangenheitsform.

Die morphologische Analyse ist üblicherweise eine Vorarbeit für das Parsen von Texten, welches in Abschnitt 2.6 genauer beschrieben ist. Dafür sind jedoch genauere syntaktische und semantische Informationen nötig. So würde beispielsweise das Wort "fishing" eindeutig durch die "-ing"-Endung als Verb

im Partizip Präsens erkannt, während bei dem Wort "fish" eine Mehrdeutigkeit zwischen dem Verb "fischen" und dem Substantiv "Fisch" die Verarbeitung verkompliziert. Dies wird durch das sogenannte Part-of-Speech-Tagging gelöst, welches in Abschnitt 2.5 behandelt wird.

4.4.1 Lexika in der Morphologie

Für eine präzise morphologische Analyse bedarf es 3 verschiedener Lexika, um mögliche Interpretationen zu speichern:

1. Liste von Affixen (zusammen mit den Informationen, die damit verbunden sind, z.B. Negation `"un-"`)
2. Liste von unregelmäßigen Wörtern (zusammen mit den Informationen, die damit verbunden sind, z.B. `"went":simple past"(to) go"`)
3. Liste von Wortstämmen (zusammen mit syntaktischer Kategorie, z.B. `"believe":verb`)

Die Affix-Liste könnte in einen Präfix- und einen Suffix-Teil getrennt werden, welche jeweils das Affix und die oben beschriebenen Informationen als formatierten Text enthalten, in etwa wie folgt (Suffix-Teil):

```
ed PAST_VERB
ed PSP_VERB
s PLURAL_NOUN
```

PAST_VERB, PSP_VERB und PLURAL_NOUN beinhalten die durch den jeweiligen Suffix hinzugefügten Informationen über das Wort.

Ähnlich verhält es sich mit der Auflistung der unregelmäßigen Formen. Das Affix wird hier durch die unregelmäßige Form ersetzt und es ist zu beachten, dass es nötig ist, zu den in 1. vorhandenen Informationen auch den Referenz-Stamm anzugeben, in etwa wie folgt:

```
began PAST_VERB begin
begun PSP_VERB begin
```

"Began" und *"begun"* sind hier jeweils die "PAST_VERB"- und "PSP_VERB"(past participle)-Form des Referenzstammes *"begin"*.

Um unnötige Fehler zu vermeiden ist es ratsam, zusätzlich ein Lexikon mit Wortstämmen zu verwenden, welches die Wortstämme und eine grammatikalische Einordnung beinhaltet. Dies hat hauptsächlich zwei Gründe:

1. Die Form im Zusammenhang mit vielen syntaktischen (siehe Abschnitt 2.5) und morphologischen Regeln ist wichtig. Die *"-able"*-Anhangsregel kann beispielsweise nur bei Verben verwendet werden, die dann ein Adjektiv darstellen (diese Eigenschaft würde in der Affix-Liste beschrieben).

2. Mögliche Wortstämme überprüfen zu können ist wichtig, da somit viele Mehrdeutigkeiten ausgeschlossen werden können. So könnte "bus" z.B. als "bu^s" (Plural von "bu") interpretiert werden. Dies ist jedoch nicht möglich, wenn kein Eintrag zu dem Wortstamm "bu" existiert.

4.4.2 Buchstabierregeln mit endlichen Zustandswandlern

In der Morphologie gibt es eine Vielzahl von Regeln, die bei der Unterteilung, Interpretation und Zusammensetzung von Wörtern zu beachten sind. Wenn man das "believe"-Beispiel aus Abschnitt 2.4 erneut betrachtet, fällt auf, dass bei dem hinzufügen des Suffixes "-able" der letzte Buchstabe des Wortstammes entfällt. Dies ist natürlich kein Tippfehler, sondern beruht auf einer der besagten Regeln. Die oben verwendete Regel lässt sich wie folgt darstellen:

$$\texttt{e} \rightarrow \varepsilon \texttt{\^{}\_able}$$

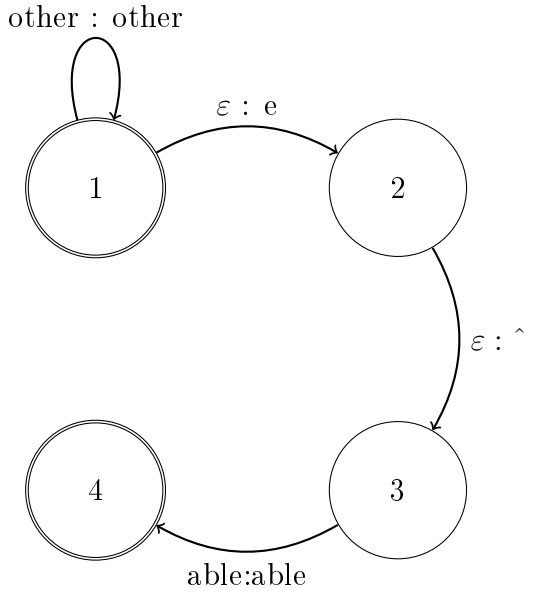
Das e ist in dem Falle der in Frage stehende Buchstabe, welcher, symbolisiert durch den Pfeil, in das ε (das leere Wort) gewandelt wird, falls ein Affix angehängt wird (^) und es an dieser Stelle (_) steht, also vor dem *"-able"*.

Solche Regeln lassen sich zu morphologischen Zwecken gut mit endlichen Zustandswandlern (finite state transducers) realisieren, was im Folgenden demonstriert werden soll.

4.4.3 Endliche Zustandswandler zur Umsetzung von morphologischen Regeln

Ein endlicher Zustandswandler ist ein Graph aus einer endlichen Anzahl von Knoten (Zuständen), welche durch Kanten verbunden sind. Knoten sind unterteilt in einen Start-Knoten, eine beliebige Anzahl an normalen Knoten und mindestens einen Ziel-Knoten. Der Start-Knoten als Input das zu verarbeitende Wort, welches nun in eine Abbildung der Aufbaustruktur des Wortes übertragen werden soll. Das Erreichen eines Ziel-Knotens heißt, dass das Wort korrekt ist und akzeptiert werden kann. Die Kanten repräsentieren

die Abbildung einer Zeichenkette auf eine andere. Dies funktioniert bidirektional, wodurch sie sowohl zum Erstellen, als auch zum Auflösen von Wörtern verwendetet werden können. Die folgende Abbildung zeigt den eindlichen Zustandswandler, der die Umsetzung der besagten "able"-Anhangs-Regel ermöglicht: (QUELLE AUTOMATEN)



Der oben abgebildete Zustandswandler ist in der Lage die beschriebene Regel umzusetzen wie folgt: In Zustand 1 erfolgt der Input des eingelesenen Wortes, also in diesem Falle *"believable"*. Über die Schleife mit *"other:other"* ist die Abbildung aller nicht speziell angegebenen Zeichenfolgen auf sich selbst beschrieben. Es kann jederzeit das leere Wort auf ein *"e"* Übertragen werden, dies geschieht jedoch nur, falls auch ein "-able" folgt. In dem Fall wird zuvor noch aus dem leeren Wort ein Affix-Zeichen (^) generiert (Es ist anzumerken, dass der Zustandswandler nicht deterministisch ist, sich jedoch über die Potenzmengenkonstruktion immer ein äquivalenter deterministischer Zustandswandler erstellen ließe). So wird also das Wort "believable" durch diesen Automaten als "believe^able" Interpretiert, und somit das "able"-Affix erkannt. Andersherum ließe sich aus dem Wortaufbau "believe^able" das korrekte Wort "believable" generieren. (QUELLE NEA VL von PREIS?)

Üblicherweise werden für jede Regel eigene Zustandswandler erstellt, welche dann bei der Verarbeitung von Wörtern parallel ablaufen. Dies ist gut für die Performanz, birgt allerdings Probleme. Wenn man mit komplexen Wörtern arbeitet, bei denen mehrere Regeln gleichzeitig anzuwenden sind, können verschiedene valide Ergebnisse erzielt werden. So wäre es etwa problematisch zwischen den Wörtern "unˆionˆiseˆed" (Chemie) und "unionˆiseˆed" (gewerkschaftlich organisiert) zu unterscheiden.

4.5 Part-of-speech tagging und Wortvorhersage

Dieses Kapitel befasst sich mit der Transformation von Ergebnissen der morphologischen Analyse einzelner Wörter in eine sinnvolle Kategorisierung der Wörter im Sachzusammenhang aus mehreren Wörtern. Part-of-speech tagging klassifiziert einzelne Worte eines Textes so, dass erfasst wird, welche Rolle sie im Satz spielen. Die einzelnen Wörter des Textes werden im Englischen bezüglich ihrer grammatikalischen Funktion und Anzahl markiert (*annotiert*). Die Identifikation muss so erfolgen, dass sie den Kontext des Satzes erfasst, also insgesamt Sinn ergibt. Diese Kennzeichnung geht über die Analyse einzelner Wörter hinaus; Sie erfasst auch die Syntax umgebender Wörter eines Satzes.

Im Deutschen ist es etwa recht wahrscheinlich, dass auf ein Nomen am Satzanfang als nächstes Wort ein Verb folgt, da dies einen typischen Satzbau darstellt. Ein part-of-speech Tagger sollte daher beispielsweise im Satz *"Ich fahre Fahrrad."* korrekt das "Ich" als Subjekt, das "fahre" als Prädikat und das "Fahrrad" als Akkusativobjekt darstellen; Die isolierte Betrachtung von *"Fahrrad"* ließe aber auch die Kategorisierung als Subjekt zu, da sich die zwei Möglichkeiten bezüglich der Morphologie nicht unterscheiden. Diese Mehrdeutigkeit einzelner Worte im Satz wird durch Analyse des syntaktischen Kontextes eliminiert, sodass POS tagging eine Eindeutigkeit der syntaktischen Deutung erzielt.

Solche Regelmäßigkeiten werden in Form von sogenannten *corpora* geliefert; sie stellen Trainingsdaten in Form von tatsächlicher Sprache dar. Im Englischen wird etwa häufig die Zeitung The Wall Street Journal genutzt, um die Repräsentation bestimmter Satzstrukturen und Wortfolgen statistisch zu erfassen. Alternativ können auch standardisierte Corpora wie der Lancaster-Oslo-Bergen Corpus (QUELLE) verwendet werden, der etwa eine Million Wörter in Sprache enthält. Corpora sind etwa für NLP auf Basis von Machine-Learning-Algorithmen erforderlich, können darüber hinaus aber

auch etwa zur Rechtschreibprüfung oder Textunterteilung benutzt werden. Die meisten Textverarbeitungsprogramme können heutzutage etwa erkennen, wenn sich syntaktische Fehler in einem Satz ereignen (Kommasetzung, falscher Kasus, etc.). Es ist anzumerken, dass part-of-speech tagging nicht unbedingt vorher annotierter Texte bedarf, aber ohne diese eine syntaktische Einordnung ungleich schwerer fällt und die Fehlerquote massiv steigt. Mehrdeutigkeiten aus der Syntax heraus können somit nicht aufgeschlüsselt werden.

Die Anwendung eines Corpus auf ein natürlichsprachliches Dokument ermöglicht sogleich auch eine Vorhersage der nächsten Wörter bzw. Sprachbausteine, nachdem schon ein Teilsatz gelesen wurde. Diese *prediction* kann auf Basis bekannter Regelmäßigkeiten also schon vor der Wortanalyse recht genau die nächste grammatikalische Komponente bzw. den Worttyp vorhersagen.

4.5.1 N-gram Modelle zur Sprachvorhersage und -modellierung

Auf Basis der vorliegenden Corpora existieren eine Vielzahl an Regeln bezüglich des Satzbaus und bekannnter Folgen von Worttypen, wenn ein neues Dokument annotiert werden soll. Zur Veranschaulichung einer möglichen Annotation soll in diesem Abschnitt die Notation [CLW7] für Annotationen genutzt werden.

Bei erneuter Betrachtung des obigen Beispiels *"Ich fahre Fahrrad."* könnte dieses, wenn dieser Satz als minimaler Corpus vorliegt, neben der Regel Subjekt-Prädikat-Objekt eines Satzbaus auch beinhalten, dass der Punkt ein Satzende markiert und das Substantive und Satzanfänge mit einem Großbuchstaben beginnen ("Fahrrad" ist gemäß der Morphologie ein Nomen und ist groß geschrieben). Sprachvorhersage nutzt diese bekannten Regeln und würde etwa aus der obigen Regel herleiten, das, wenn ein Verb gelesen wurde, darauf immer ein Substantiv bzw. das Satzobjekt folgt. POS-Tagging macht sich diese Beobachtungen durch Berechnung von Wahrscheinlichkeiten bei der Annotation von Wortfolgen zunutze.

Kontextfreie Betrachtung einzelner Wörter, also ohne andere Taggings mit einzubeziehen, wird als Unigramm bezeichnet. Dabei werden jedoch Regeln über den Zusammenhang von Wortketten vollständig ignoriert und das Ergebnis ist ähnlich dem der morphologischen Analyse. Vorhersage-Systeme, die sich der Annotierung des vorigen Wortes bedienen, werden als Bigramme (englisch *bigrams*) bezeichnet, Trigramme dementsprechend bei

Berücksichtigung der zwei vorigen Wörter und N-gramme, wenn alle n-1 zuvor gelesenen Wörter, also die gesamte Terminologie des Textes bis zum n-ten Wort, berücksichtigt werden. Die Wahrscheinlichkeit des Typs des Folgewortes auf Basis eines Bigramms lässt sich berechnen, indem aus einem Corpus die Worttypen an der jeweiligen Satzstelle gezählt und statistisch erfasst werden. Als Beispiel dazu soll hier zunächst ein Bigramm zur tatsächlichen Wortvorhersage dienen, anschließend wird erläutert, inwiefern solche Techniken auch den nächsten part-of-speech in einem Text vorhersagen können. Aus folgendem Corpus vierer Aussagen lässt sich dies wie folgt ableiten:

```
<s> ich mag Züge <s> ich mag guten Tee
<s> guten Morgen <s> guten Abend <s>
```

Zwischen einzelnen Äußerungen wird der Platzhalter <s> als Indikator einer neuen Aussage verwendet, sodass dem Bigramm nun eine Zeichenkette mit den obigen Aussagen getrennt durch `<s>` zur Verfügung steht. Nun wird die mögliche Wortkombinatorik aus dem Corpus extrahiert, indem Kombinationen gezählt werden. Berechnet werden nun die Wahrscheinlichkeiten des Folgewortes auf Basis der vorherigen, formal ausgedrückt: $\frac{C(w_{n-1}w_n)}{\sum wC(w_{n-1}w)}$ Es

Sequenz	Anzahl	$P(w_n\|P(w_{n-1})$
<s>	4	
<s> ich	2	0,50
<s> guten	2	0,50
ich mag	2	1,0
mag Züge	1	0,5
mag guten	1	0,5
Züge	1	
Züge <s>	1	1
guten	3	
guten Tee	1	$\frac{1}{3}$
guten Morgen	1	$\frac{1}{3}$
guten Abend	1	$\frac{1}{3}$
Tee	1	
Tee <s>	1	1
Morgen	1	
Morgen <s>	1	1
Abend	1	
Abend <s>	1	1

Tab. 1: Wahrscheinlichkeitstabelle Bigram zur Wortvorhersage

lassen sich nun die Wahrscheinlichkeiten des jeweils nächsten Wortes nach

einem gelesenen Wort ablesen. Auf "ich mag" folgt insgesamt einmal "Züge" und einmal "guten". Daher kann nun genau die Wahrscheinlichkeit des nächsten Wortes "Züge" mit $P(\text{"Züge"}|\text{"ich mag"}) = 50\%$ berechnet werden.

4.5.2 Stochastisches POS-tagging

Neben der Möglichkeit, einzelne Wörter vorherzusagen, eignen sich n-Gramme wie oben erwähnt auch zur Darstellung der Wahrscheinlichkeiten von Satzbausteinen, also der von Platzhaltern für alle möglichen Wörter einer Kategorie bzw. Komponente. Stochastisches Part-of-speech tagging kann anhand dieser die Wahrscheinlichkeiten vorhersagen, welcher Worttyp folgen wird. Die Trainingsdaten sind, nach Best-Practices aus dem maschinellen Lernen, in der Regel manuell und durch Benutzer markiert worden, damit dem Programm ein korrekter Datensatz zur Verfügung steht. Auf Basis eines Corpus können somit Mehrdeutigkeiten durch den syntaktischen Zusammenhang eindeutig aufgelöst werden. In einem Beispiel mit syntaktischer Ambiguität werden jedoch Wörter zunächst verschieden annotiert. Das tagset von CLAWS 7 annotiert Wortkategorien wie folgt:

PPIS2	1st person plural subjective personal pronoun
PPHO1	3rd person sing. objective personal pronoun
NN1	singular common noun
VV0	base form of lexical verb
VVD	past tense of lexical verb
PUN	punctuation

Tab. 2: Annotationen gemäß [CLW7] (Auszug)

Als Beispiel dient hier der englische Satz *"We saw her duck."*. Dieser enthält Mehrdeutigkeiten, da "duck" entweder als *"Ente"*, also als Substantiv *"ihre Ente"* oder als "ducken", also als Verb (dt. *"sie (sich) ducken"*) interpretiert werden kann. Zusätzlich kann "we saw" als *"sahen"*, oder als *"zerschneiden"*, also zeitformabhängig verstanden werden. Äquivalent im Deutschen sind also die drei Sätze *"Wir sahen ihre Ente."*, *"Wir sahen sie (sich) ducken."* und *"Wir zerschneiden ihre Ente"* mit völlig unterschiedlicher Aussage. (Semantisch betrachtet stellt sich ferner die Frage, ob die Deutung von "her duck" als Nomen mit "ihr Haustier" etwas anderes bedeutet als "für sie zum Essen")

Diese sinnvollen Annotationen könnten durch einen CLAWS 7-POS-tagger wie folgt erfasst werden:

1. `We(PPIS2) saw(VVD) her(PPHO1) duck(VV0) .(PUN)`
2. `We(PPIS2) saw(VVD) her(PPHO1) duck(NN1) .(PUN)`
3. `We(PPIS2) saw(VV0) her(PPHO1) duck(NN1) .(PUN)`

Problematisch ist, dass auch syntaktisch falsche Deutungen entstehen können, wenn, wie oben erwähnt, unzureichende Trainingsdaten und Regeln für die Wortfolgen existieren. Dann werden alle möglichen Teildeutungen der einzelnen Wörter als Tagging ausgegeben, etwa mit der Bedeutung *"Wir zerschneiden sie sich ducken"* aus der Syntax.

4. `We(PPIS2) saw(VV0) her(PPHO1) duck(VV0) .(PUN)`

Durch das Verwenden von manuell erstellten Corpora kann zumindest diese eindeutig falsche Deutung der Syntax eliminiert werden, da solch ein Sprachkonstrukt in der englischen Sprache nicht auftreten kann. Das Tagging des folgenden minimalen Korpus könnte dann wieder als Eingabe für ein n-Gramm dienen, welches auf Basis der höchsten Wahrscheinlichkeit ein Wort annotiert.

`People use to see a duck. Most people duck in order to avoid danger.`

Der Corpus könnte annotiert sein:

`We(NN2) use(JK) to(TO) see(VV0) a(AT1) duck(NN1).(PUN) Most(DAT) people(NN2) duck(VV0) in(BCL11) order(BCL12) to(BCL13) avoid(VV0) danger(NN1).`

Dauraus ergibt sich unter Verwendung eines n-Grams umgangssprachlich formuliert, dass etwa in einem Satz, in welchem bereits ein Wort als Verb im Präsens getaggt wurde, ein Objekt, also ein als Nomen getaggtes Wort, folgen muss. Im ersten Satz ergibt sich dies etwa aus der Stellung des "see" als Verb vor dem "duck". Dementsprechend kann so schon die vorhin beschriebene Deutung 4. ausgeschlossen werden, da diese Regel des Satzbaus nicht vorkommt. Weitere mögliche Regelmäßigkeiten, die ein n-Gramm statistisch erfasst, wäre die Möglichkeit einer Before-Clause (BCL) nur nach einem Nomen oder die hohe Wahrscheinlichkeit eines Punktes nach einem Nomen im Singular. Auffällig ist, dass die Deutung von saw als Verb in Past Tense entfällt, da keine Regel auftritt, die ein Verb in Past-Tense vor einem Nomen enthält. Daher erfasst ein N-Gram auf Basis dieses Korpus nur Verben im Präsens korrekt (im Corpus taucht keine Deutung eines Verbs als Vergangenheitsform auf).

Abschließend lässt sich daher sagen, dass die Trainingsdaten möglichst repräsentativ für die möglichen Konstruktionen in einer Sprache bzw. in der Anwendungsdomäne des POS-taggers sein müssen, um akkurate Ergebnisse

zu erzielen. Dies betrifft sowohl die Qualität der im Corpus enthaltenen Wörter und Sätze, als auch deren Korrektheit im Hinblick auf die quantitative Repräsentation einzelner Konstrukte. Häufig anzutreffende Hauptsatz-Nebensatz-Konstruktionen müssen etwa für die Analyse eines stilistisch einfachen Textes, der zum Großteil aus solchen besteht, häufiger vorhanden sein, als kompliziert verschachtelte Kausalsätze und eingeschobene Relativsätze mit beispielsweise vielen beschreibenden Aufzählungen.

4.6 Parsen und Generieren mit kontextfreien Grammatiken

Wie bereits in 2.4 angekündigt befasst sich dieser Abschnitt mit dem Parsen und Generieren von Sätzen auf Grundlage der vorhergegangenen Verarbeitungsschritte. Fokus liegt dabei auf der Auflösung von Mehrdeutigkeiten, welche nicht durch die morphologische Analyse und Part-of-Speech-Tagging bewältigt werden konnten. Dies wird anhand verschiedener kontextfreier Grammatiken genauer erläutert.

4.6.1 Generative Grammatik

Der Begriff der generativen Grammatik beruht auf der Arbeit von Chomsky in den 1950er Jahren [CHO57]. Man versteht darunter eine formale Grammatik, welche in der Lage ist alle möglichen Sätze einer natürlichen Sprache und nur genau diese zu generieren. Hier ist anzumerken, dass man dem tatsächlichen Entwurf einer solchen Grammatik bisher nicht einmal nahe gekommen ist. Jedoch ist für Linguisten der prinzipielle Aufbau von solchen Grammatiken höchst interessant (vor allem von solchen Grammatiken, welche sich auf alle natürlichen Sprachen anwenden lassen), während sich NLP-Forscher eher mit dem Bau und der Nutzung von möglichst umfangreichen Grammatiken befassen.

Ähnlich zu der ableitenden Morphologie versucht man hier, Zeichenketten eine interne Struktur zu verleihen, welche es ermöglicht, Informationen zu den einzelnen Bestandteilen zu extrahieren, jedoch tut man dies mit Sätzen (oder Satzteilen) anstelle von einzelnen Wörtern. Darstellen lässt sich diese Struktur beispielsweise durch Klammerung wie folgt:

((the (fisherman)) fishes) ((the (old fisherman)) fishes)

Durch das obige Beispiel wird die Bindung der einzelnen Wörter untereinander deutlich: "old" gehört eindeutig zu "fisherman" und das "the"

bezieht sich im zweiten Satz auf "old fisherman" zusammen. Die Klammerung (the old) ist also auf keinen Fall möglich, da "old" mit dem "fisherman" eingeklammert ist.

Bei zwei Grammatiken spricht man von schwacher Äquivalenz, falls sie die gleichen Zeichenketten generieren können und von starker Äquivalenz, wenn sie diese auch gleich klammern.

Die meisten Ansätze verleihen der internen Struktur nun verschiedene Bezeichnungen, abhängig von der oben gezeigten Klammerung. Somit würde man "the fisherman" (ebenso wie "the old fisherman") im Englischen als "noun-phrase" (NP) bezeichnen. "fishes" würde (zusammen mit jeglichen Zusätzen wie "fishes very good" oder "fishes fish" als "verb-phrase" (VP) gekennzeichnet. Diese Bezeichnungen dienen in kontextfreien Grammatiken als linke Seite von Produktionsregeln, wie im folgenden Abschnitt genauer dargestellt wird.

4.6.2 Kontextfreie Grammatiken

Kontextfreie Grammatiken stammen aus dem Bereich der Theoretischen Informatik (QUELLE PREIS) und werden dargestellt durch ein Tupel mit 4 verschiedenen Komponenten:

1. Eine Menge nicht-terminaler Symbole (wie z.B. NP oder VP aus dem vorherigen Abschnitt)

2. Eine Menge terminaler Symbole (die Wörter aus denen später die Zeichenkette bestehen wird, also "the", "old", "fisherman" und "fishes")

3. Eine Menge sogenannter Produktionsregeln mit folgenden Eigenschaften:

   (a) Eine Regel besteht aus einer linken und einer rechten Seite, getrennt durch einen Pfeil ($\rightarrow$).

   (b) Auf der linken Seite befindet sich genau ein nicht-terminales Symbol.

   (c) Auf der rechten Seiten befindet sich mindestens ein Symbol (terminal oder nicht-terminal).

4. Das Startsymbol (konventionell "S"), welches zu den nicht-terminalen Symbolen gehört

Es ist zu beachten, dass in kontextfreien Grammatiken generell das leere Wort ε als terminales Symbol mit eingeschlossen ist. Aus linguistischer Sicht verkompliziert dies jedoch nur die Grammatiken und kann daher bezüglich NLP ausgeschlossen werden (es ließe sich außerdem auch zu jeder kontextfreien Grammatik eine schwach-äquivalente Grammatik erzeugen, welche das leere Wort nicht beinhaltet).

Man spricht von einer linksbündigen bzw. rechtsbündigen Grammatik, wenn alle nicht-terminalen Symbole ganz links bzw. rechts auf der rechten Seite der Regel stehen (also in der Form "NT→NT t" bzw "NT→t NT"). Diese Grammatiken sind jedoch schwach-äquivalent zu regulären Grammatiken, welche sich mit endlichen Automaten implementieren lassen. Dies ist jedoch für den Aufbau von natürlichen Sprachen nicht ausreichend. Der Ansatz, sich auf eine rechts- bzw. linksbündige Grammatik zu beschränken, um damit eine natürliche Sprache zu generieren, entfällt also sofort.

Im folgenden wird die Funktionsweise kontextfreier Grammatiken anhand eines sehr simplen Beispiels dargestellt. Hierzu gelten die folgenden Produktionsregeln P (für spätere Erläuterungen durchnummeriert):

1. `S` → `NP VP`
2. `VP` → `V`
3. `VP` → `V NP`
4. `VP` → `V VP`

Außerdem stehe ein Lexikon die folgenden Wörter zur Verfügung:

`V` → `can`
`V` → `fish`
`V` → `catch`
`NP` → `fish`
`NP` → `he`
`NP` → `fishermen`

Die formale Darstellung dieser Grammatik sieht dann aus wie folgt:

$$G = (\{S, VP, NP, \}, \{can, fish, catch, he, fisherman\}, \{P(sieheoben)\}, S)$$

Man beachte, dass "fish" nur einmal bei den terminalen Symbolen auftaucht, obwohl es im Lexikon zwei mal aufgeführt ist (als verb und als nounphrase). Diese Unterscheidung wird in der Grammatik jedoch lediglich durch

die produzierende Regel vorgenommen, das terminale Symbol ist für beide Varianten zunächst gleich.

Die Möglichkeiten und Probleme der beschriebene Grammatik werden an den folgenden Beispielsätzen deutlich:

```
"He can fish" "He can catch fish" "Fishermen can fish fish"
```

Klammert man den ersten Satz entsprechend der Grammatik kommt man zu folgendem Ergebnis:

```
(S(NP he)(VP(V can)(VP(V fish))))
```

Somit kann man, indem man die Klammern von außen nach innen durchgeht die verwendeten Regeln und ihre Reihenfolge ermitteln, in diesem Fall also 1., 4. und dann 2.. (Dies lässt sich jedoch sehr viel übersichtlicher darstellen, wie in einem späteren Abschnitt aufgezeigt wird.)

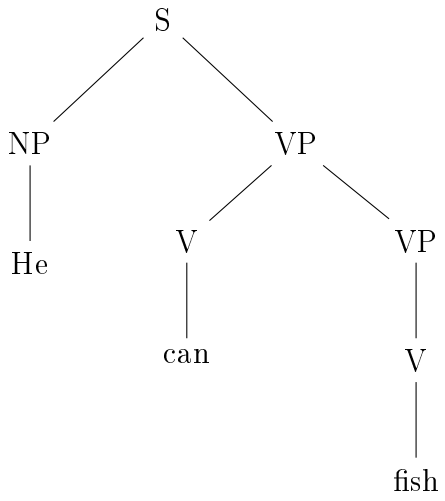
Der gleiche Satz ließe sich aber auch leicht anders klammern:

```
(S(NP he)(VP(V can)(NP fish)))
```

Die Grammatik lässt in diesem Fall durch die 3. Produktionsregel zu, dass "fish" als noun-phrase statt verb interpretiert wird , was jedoch auf semantischer Ebene keinerlei Sinn ergibt und daher eigentlich nicht möglich sein sollte. Entfernt man die 3. Produktionsregel jedoch aus der Grammatik, so lassen sich andere, durchaus sinnvolle Sätze, wie der zweite Beispielsatz "He can catch fish" nicht mehr herstellen. Dies gilt ebenfalls für den dritten Beispielsatz "Fishermen can fish fish", welcher nur Sinn ergibt, solange "fish" als verb und als noun-phrase interpretiert werden kann.

4.6.3 Pars-Bäume

Wie angekündigt lässt sich die Struktur von geparsten Sätzen auch etwas übersichtlicher darstellen. Dies geschieht mit sogenannten Pars-Bäumen. Ein Pars-Baum zu dem oben genannten Beispiel sähe beispielsweise aus wie folgt:



Da auf der linken Seite einer Produktionsregel der kontextfreien Grammatiken immer nur ein nicht-terminales Symbol steht, lässt sich hier unschwer eine Regel pro Elternknoten erkennen. Das S wir in NP und VP aufgeteilt, das NP wird widerum zum terminalen Symbol *"He"* usw. entsprechend der definierten Produktionsregeln. Alle Blätter (Knoten ohne Kindknoten) des Baumes sind jeweils ein Wort und bilden somit zusammen den erzeugten Satz *"He can fish."*.

4.6.4 Zufallsgenerierung mit kontextfreien Grammatiken

Ähnlich zu den endlichen Zustandswandlern sind auch kontextfreie Grammatiken bidirektional und können somit nicht nur zum Parsen, sondern auch für die Generierung von Sätzen verwendet werden. Ein naiver Zufallsalgorithmus könnte dabei auf einer Liste ausgeführt werden und die folgenden Schritte verwenden:

1. Wähle ein nicht-terminales Symbol aus der Liste (zufällig oder z.B. der Reihenfolge nach).

2. Wähle zufällig eine Produktionsregel der Grammatik mit dem gewählten Symbol auf der linken Seite und führe sie aus.

3. Ersetze in der Liste das gewählte Symbol durch die rechte Seite der gewählten Produktionsregel

4. Wiederhole 1., 2. und 3. bis keine nicht-terminalen Symbole übrig sind.

5. Durchlaufe die Liste und speichere die Inhalte der Listenelemente in eine gemeinsame Zeichenkette. Diese Zeichenkette ist der Zufällig generierte Satz.

Mit dem beschriebenen Verfahren lassen sich zweifellos Sätze generieren, jedoch können diese sowohl Fehler enthalten, als auch in vielen Fällen unendlich lang werden. Dies lässt sich beides mit der in 2.5.3 beschriebenen Grammatik demonstrieren.

Fehlerhafte Generierung: Wegen der Bidirektionalität der kontextfreien Grammatik, geht mit der gezeigten Möglichkeit falsch zu interpretieren auch die Möglichkeit einher falsch zu generieren. Man beachte z.B., dass sich der fehlerhafte Satz "He fish can" genauso generieren ließe wie der korrekte Satz "He can fish".

Unbegrenzte Sätze: Wie in der natürlichen Sprache lassen sich theoretisch unendlich lange Sätze formulieren. Dies wird normalerweise durch den Menschen unterbunden, kann jedoch einen Zufallsalgorithmus abhängig von den verwendeten Regeln für lange Zeit am terminieren hindern. So kann in der verwendeten kontextfreien Grammatik schon allein durch die 4. Produktionsregel eine unendlich lange Folge von Verben generiert werden.

4.6.5 Anwendung Chart-Parsing

Wie anhand des Parsbaums zu einer zugehörigen Grammatik erkennbar ist, ist es nicht nur möglich, anhand von Produktionsregeln Sätze zu generieren, sondern diese (Teil-)Sätze aufsteigend nach den erzeugenden Regeln aufzuschlüsseln. Aus kontextfreien Grammatiken lassen sich jedoch Sätze erzeugen, die semantisch betrachtet weniger Sinn ergeben, als andere. Auch kann es passieren, dass man in eine Sackgasse läuft, da keine passende Produktionsregel mehr existiert, und dann backtracking erforderlich ist. Der Einsatz und die Konstruktion eines Parsers, der diese Besonderheiten berücksichtigt, werden im Folgenden anhand des bekannten Beispielsatzes *they can fish* erläutert.

Da die Anwendung der Produktionsregeln hintereinander erfolgt, muss daher festgehalten werden, welche Regeln bisher zum Einsatz gekommen sind. Für die Implementierung ist diese Speicherung zur Reduzierung der Programmlaufzeit erforderlich. Anhand dieser bekannten Informationen kann dann rekursiv mithilfe von dynamischer Programmierung die Generation von

Sätzen erfolgen und alle möglichen Regeln werden erfasst. Es lässt sich daran ein Suchbaum für den Einsatz der Grammatik ableiten.

Dieser bottom-up Ansatz repräsentiert die einzelnen Wörter eines Satzes als Knoten und fügt sukzessive die Regeln zwischen den möglichen Satzteilen in Form von Kanten hinzu. Kanten stellen die bis dahin angewandten Produktionsregeln dar, indem sie diese als $mother_{category}$ enthalten und können verschieden weit entfernte Knoten links und rechts verbinden. Kanten besagen also, als welche grammatikalischen Komponenten die verbundenen Knoten verstanden werden können. Gemäß [COP04] besteht eine Kante aus:

$[id, knoten_{links}, knoten_{rechts}, mother_{category}, daughters]$

Die Grundlage für den Algorithmus sind dann die Knoten zwischen Wörtern, im Beispiel:

Initialisiert wird mit einer leeren Menge von Kanten zwischen den bekannten Knoten. Der Algorithmus terminiert, wenn eine Kante zwischen Anfangs- und Endknoten gefunden wurde. Dies bedeutet, dass alle dazwischen liegenden Teilregeln ebenfalls gefunden wurden; der Suchbaum ist dann vollständig. Nach dem Durchlauf liegen dann alle möglichen Kanten, das heißt alle Teil- und Gesamtdeutungen des Satzes vor. Begonnen wird mit mit den ersten Knoten; diese werden dann mittels Kanten direkt nach den Regeln der Grammatik verbunden und nach und nach kommen weitere Kanten bis zum n-ten Wort hinzu. Wenn mehrere Kanten von erstem zu letztem Wort existieren, deutet dies dann auf eine Mehrdeutigkeit hin, da die Kanten dann verschiedene Produktionsregeln enthalten. Der Algorithmus sieht dann wie folgt aus:

1. `ChartParse:`
   `for each Wort` w_{0-n} `im Satz, sei` *`from`* `der linke`
   `und` *`to`* `der rechte Knoten und daughters ds ein w`
   `for each möglichen Lexikoneintrag der Kategorie cat von w`
   `neueKante` *`from,to,cat,ds`*
   `output alle aufgespannten Kanten von` w_0 `nach` w_n
   (gesamter Text wird abgedeckt und mother ist S)

2. `neueKante:` *from, to, cat, daughters*
   `Neue Kante einzeichnen:` *id, from, to, cat, ds*
   `for each Regel linkeSeite ->` r_1–r_n `einer cat aus der Grammatik`
   `Finde zusammenhängende Menge von Kanten`
   $[id_1, from_1, to_1, cat_1, ds]$ `bis` $[id_{n-1}, from_{n-1}, from, cat_{n-1}, ds_{n-1}]$
   (so einzeichnen, dass die Kante jeweils eine bekannte Regel
   zur Verbindung zwischen zwei Knoten enthält, also $to_1 = from_2$ usw.)
   `For each Menge von Kanten, neueKante` $from_1, to, linkeSeite, id_1$–$id$ (also alle Kanten weiter unterteilen)

Bei Eingabe des Beispielsatzes würde die Ergebnistabelle aus [COP04] folgende Kanten als Einträge beinhalten:

4.6.6 Mängel kontextfreier Grammatiken

Die Umsetzung der Modellierung von Sprache auf Basis kontextfreier Grammatiken ist mit einigen Problemem verbunden. Zunächst scheint es vorteilhaft, wenn durch die Rekursion alle möglichen Teildeutungen erfasst werden; davon sind jedoch nicht alle sinnvoll und viele redundant, weil sinnlos. Es fällt auf, dass durch die Zerlegung in Teildeutungen durch Anwendung aller möglichen Regeln aus der Grammatik prinzipiell exponentiell viele Deutungen entstehen könnten. Teilweise würden auch grammatikalisch falsche Sätze akzeptiert, wenn nicht jeder Numerus, Kasus etc. eines Wortes als Terminalsymbol in der Menge an Produktionsregeln repräsentiert wird. In der obigen einfachen Grammatik wird *"they fish"* genau so wie *"it fish"* akzeptiert.

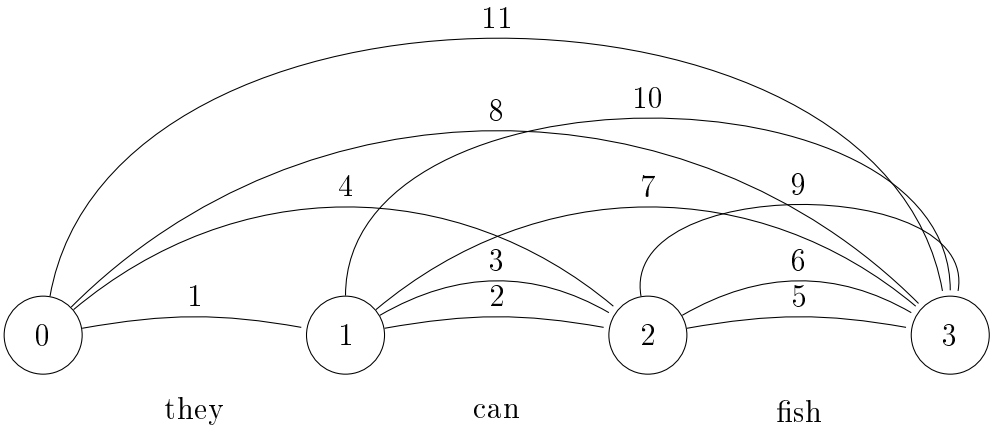
Ferner werden Zusammenhänge aus Wörtern, die sich inhaltlich aufeinander beziehen, nicht abgebildet und nicht aus dem Kontext erfasst. Manche Verben erscheinen etwa ohne Berücksichtigung des Satzobjektes für den Menschen eher sinnlos, *"Felix erwartet"* scheint unvollständig, die richtige Deutung von Zusammenhängen *"Felix erwartet Aaron"* schließt den Kontext des Teilsatzes ein.

Eine Möglichkeit zur Lösung wird in Abschnitt 2.6 und 2.7 behandelt, indem der Konsens einer Formulierung ebenso wie Ansätze der Semantik bei der Deutung berücksichtigt werden. Dies ist mittels einfacher Produktionsregeln nicht möglich.

id	$knoten_{links}$	$knoten_{rechts}$	*mother*	*daughters*
1	0	1	NP	(they)
2	1	2	V	(can)
3	1	2	VP	(2)
4	0	2	S	(1 3)
5	2	3	V	(fish)
6	2	3	VP	(5)
7	1	3	VP	(2 6)
8	0	3	S	(1 7)
9	2	3	NP	(fish)
10	1	3	VP	(2 9)
11	0	3	S	(1 10)

Tab. 3: Parsing-Tabelle gemäß Grammatikregeln aus [COP04]

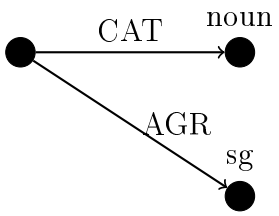
Wie sich nun erkennen lässt, lässt sich anhand der mit S beginnenden Einträge aus der Tabelle der Satz generieren, indem nacheinander die Produktionsregeln aus jder Zeile angewendet werden. Es lässt sich jede mögliche Deutung ablesen, die die Grammatik zulässt. Der Algorithmus gibt als Ergebnis also alles möglichen Parsbäume in der Tabelle an. Das Ergebnis kann wie folgt visualisiert werden (Ein schrittweises Beispiel findet sich in der Arbeit von Ann Copestake in Kapitel 4.8):



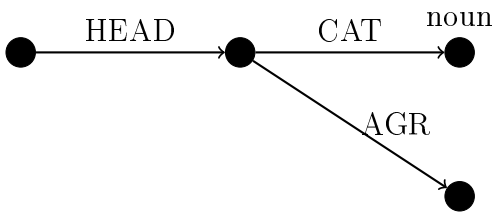
4.7 Parsen mit constraint-basierten Grammatiken

Wie zuvor an verschiedenen Beispielen demonstriert wurde, weisen die kontextfreien Grammatiken beim Parsen und Generieren einige schwächen auf. Daher wird im folgenden Abschnitt ein neuer, komplexerer Ansatz vorgestellt um sich mit den gleichen Problemen zu befassen. Kontextfreie Grammatiken können als constraint-basiert interpretiert werden, diese Bezeichnung wird jedoch üblicherweise für mächtigere Grammatiken verwendet. Constraint-basierte Grammatiken sind in NLP weit verbreitet und werden mit sogenannten feature-structures (Eigenschaftsstrukturen) dargestellt, daher auch oft feature-structure (FS) Grammatik genannt. Im folgenden Abschnitt soll die Umsetzung solch einer Grammatik durch feature-structures demonstriert werden.

FSs sind Graphen, welche große Ähnlichkeit zu Baumstrukturen aufweisen. Sie haben genau eine Wurzel und sind von dieser aus gerichtet. Ein Knoten kann keinen, einen oder mehrere Kindesknoten besitzen. Endknoten, also Knoten mit dem Ausgangsgrad 0 verfügen über Werte und Kanten sind mit Eigenschaften versehen. Ein Weg entlang von Eigenschaften wird als Pfad bezeichnet. Eigenschaften sind entweder atomar oder zusammengesetzt, abhängig davon ob sie auf einen Endknoten oder auf einen weiteren nicht-Endknoten zeigen. In dem folgenden, dem Paper entnommenen Beispiel ist eine FS abgebildet, die sich in zwei Eigenschaften, *"CAT"* für die Wortkategorie und *"AGR"* für agreement (Stimmigkeit), teilt, welche jeweils auf die Endknoten mit den Werten *"noun"* (für Nomen) und *"sg"* (für Singular) zeigen. Die Eigenschaften *"CAT"* und *"AGR"* sind also atomar.



Fügt man wie in der nächsten Abbildung vor der Wurzel eine weitere Eigenschaft *"Head"* hinzu, so ist diese zusammengesetzt, da sie auf einen nicht-Endknoten zeigt. Dieser wird wiederum in die beiden Eigenschaften *"CAT"* und *"AGR"* aufgeteilt.



Solche Strukturen lassen sich am besten als attribute-value matrices (Attribut-Wert Matrizen, AVMs) aufschreiben was dann wie folgt aussieht:

In FSs ist es möglich, dass mehrere Pfade zu dem gleichen Endknoten erreichen, was als Eintrittsinvarianz bezeichnet wird. Sollte dies bei einem Endknoten zutreffen, so wird dessen Wert durch einen eingeklammerten Integer dargestellt. Der numerische Wert dieses Integers ist irrelevant, da er als Platzhalter für einen tatsächlichen grammatikalischen Wert verwendet wird. Dabei ist zu beachten, dass dann für gleiche Integer auch der gleiche Wert eingesetzt wird.

In der folgenden Abbildung sind Eintrittsinvarianz und die Darstellung von FSs als AVMs dargestellt:

Üblicherweise werden bei dem Umgang mit FSs viele zunächst simple FSs zu größeren Vereint, um später aus einzelnen Wörtern korrekte, zusammenhängende Sätze oder Satzteile zu bilden. Diese Vereinigung (oder *unification*) erlaubt es alle Informationen der einzelnen FSs zu kombinieren. Leere eckige Klammern ([]) stehen dabei für einen bisher nicht spezifizierten Wert, welcher später gewählt und dann konsistent verwendet werden muss. Das genaue Vorgehen soll an dem folgenden Beispiel veranschaulicht werden:

Die unten beschriebe constraint-basierte Grammatik fügt zusätzlich zu den Wortkategorien in der in 3.6.2 verwendeten Grammatik das *agreement* zu den Produktionsregeln hinzu. Somit wird nicht nur auf die entsprechende Kategorie wie Verb oder Nomen, sonder auch auf die Multiplizitäten *singular* und *plural* geachtet.

```
S → NP-sg VP-sg
S → NP-pl VP-pl
VP-sg → V-sg NP-sg
VP-sg → V-sg NP-pl
VP-pl → V-pl NP-sg
VP-pl → V-pl NP-pl
```

Dementsprechend muss nun auch das verwendete Lexikon (als Produktionsregeln dargestellt) angepasst werden:

```
NP-sg → he
NP-pl → fishermen
NP-sg → fish
NP-pl → fish
VP-sg → catches
VP-pl → catch
VP-sg → fishes
VP-pl → fish
```

Mit den vorgestellten Änderung wir nun beispielsweise ein Satz wie *"He catch fish"* nicht mehr akzeptiert, da *"He"*(singular) und *"catch"*(in diesem Fall plural) über die ersten beiden Produktionsregeln an die gleichen Multiplizitäten gebunden sind. Diese lässt sich in der Darstellung der Grammatik durch FSs besonders gut erkennen:

4.8 Lexikalische Semantiken und kontextbasierte Deutungen

References

[COP04] Ann Copestake, University Of Cambridge, "Natural Language Processing", 2004

[CLW7] University Centre for Computer Corpus Research on Language, Lancaster University, "CLAWS part-of-speech tagger for English", http://ucrel.lancs.ac.uk/claws/, abgerufen am 19.08.2018 16.43 Uhr

[CHO57] Noam Chomsky, "Syntactic Structures", Mouton & Co., Feb. 1957