Universität Zürich
Sommersemester 2000

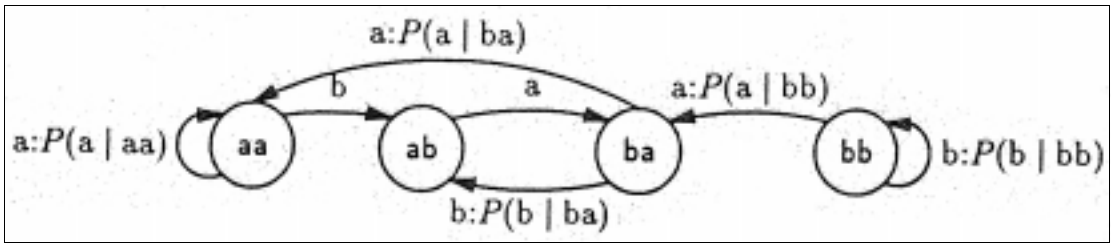
Seminararbeit

Seminar:
Syntaxtheorien und computerlinguistische Praxis

Dozenten:
Prof. Dr. M. Hess lic. phil. S. Clematide lic. phil.G. Schneider

Trigrams'n'Tags

Methoden der Korpusanalyse und des statistischen Parsings in NEGRA



Christoph Biveroni
Wiesenstr. 19
8307 Effretikon
Tel.: 052-343 15 23
E-mail: ChristophBiveroni@access.unizh.ch Juni 2000

0 Inhaltsverzeichnis

Die vorliegende Arbeit gliedert sich in drei Teile. Zuerst wird das zentrale theoretische Konzept hinter *NEGRA*, die Ausrichtung auf sprachliche Performanzdaten, erörtert. Der zweite Teil schildert sehr knapp, wie die sprachlichen Daten in *NEGRA* repräsentiert werden, und der dritte, titelgebende und ausführlichste Teil beschäftigt sich mit deren Gewinnung und Verarbeitung.

1 Einführung in die Performanzlinguistik

1.1 Kompetenz versus Performanz

Anknüpfend an de Saussures Unterscheidung von *langue* und *parole* postulierte Chomsky in seinem Werk „Aspects of the Theory of Syntax" (1965) die Dichotomie zwischen „einer allgemeinen Sprachfähigkeit und der individuellen Sprachverwendung"[1], zwischen *Kompetenz* und *Performanz*.

Das Ziel der von Chomsky begründeten *generativen Transformationsgrammatik* besteht darin, ein Sprachmodell zu konzipieren, das die aus der Kompetenz resultierende Sprachfähigkeit eines idealen[2] Sprechers möglichst adäquat abbildet (und, hier zweitrangig, eine „angemessene Hypothese über den Spracherwerb"[3] darstellt). Der kompetente Sprachbenutzer ist dabei diejenige Instanz, die implizite Sprachstrukturen explizit macht, die z.B. implizit repräsentiertes sprachliches Wissen in kontextfreie *Phrasenstrukturregeln* oder *Transformationsregeln* überführt. Zwangsläufig stellt sich bei solchem Vorgehen das Problem der Idealisierung. Im selben Masse als sich der Sprecher in seiner individuellen Sprachverwendung vom Ideal dessen, wozu er eigentlich gemäss dem Kompetenz-Modell fähig sein müsste, entfernt, verringert sich auch die Beschreibungsadäquatheit des deduktiv erstellten Grammatik-Modells (im Hinblick auf das tatsächlich produzierte Sprachmaterial).

Für die automatische bzw. maschinelle Sprachverarbeitung kann sich dieser Umstand verheerend auswirken. Wird das theoretische Regelwissen einer Sprache ohne die Berücksichtigung von sprachlichen Performanz-Phänomenen, die sich eben gerade durch Abweichungen von der regelmässigen Wohlgeformtheit und durch Unvollständigkeiten auszeichnen, zur Sprachanalyse verwendet, so leidet die Robustheit des Systems beträchtlich unter dieser Inadäquatheit. Das (nicht gerade repräsentative) Beispiel eines *DCG-Parsers* zeigt, dass das System zwangsläufig an einer ihm unbekannten Syntaxkonstruktion scheitern muss: sei es aufgrund der Nicht-Wohlgeformtheit, sei es aufgrund der Unvollständigkeit eines Satzes – die Analyse wird

[1] Bussmann 1990, 396.
[2] „Ideal" heisst hier soviel wie „einer homogenen, von dialektalen oder soziolektalen Sprachvarianten freien Sprachgemeinschaft angehörig". Vgl. Bussmann 1990, 801-803.
[3] Bussmann 1990, 396.

fehlschlagen, weil in der zugrunde liegenden Grammatik kein entsprechender Regeleintrag vorliegt.

1.2 Performanzlinguistik am Beispiel NEGRA

In der Folge soll das Projekt *NEGRA* kurz vorgestellt und insbesondere der Aspekt der Performanzlinguistik hervorgehoben werden.

1.2.1 Das Projekt NEGRA

NEGRA[4] stellt einen Teil des interdisziplinären[5] Forschungsprojektes *Sonderforschungsbereich 378* dar, das sich der Untersuchung „ressourcenadaptiver kognitiver Prozesse" widmet, 1996 als Stiftung der *Deutschen Forschungsgemeinschaft* gestartet wurde und an der Universität des Saarlandes beheimatet ist.
Die Bedeutung ressourcenadaptiver kognitiver Prozesse liegt, wie der Begriff bereits andeutet, in ihrer Eigenschaft, sich an die situativ gesteckten Grenzen kognitiver Verarbeitungskapazität anzupassen. Auf den Bereich der Informatik angewandt, heisst dies z.B., Software zu entwickeln, die je nach verfügbarem Arbeitsspeicher oder verfügbarer Rechenzeit „skalierbar" ist, d.h. in ihrer Leistungsfähigkeit und/oder ihrem Leistungsumfang an veränderte Rahmenbedingungen angepasst werden kann.

1.2.2 NEGRA: Kompetenz und Performanz

Ein grundsätzliches Bestreben der Computerlinguistik besteht darin, sprachliche Datenverarbeitung in „real-time" zu ermöglichen bzw. Analyse und Generierung von natürlichsprachlichen Phänomenen in möglichst kurzer Rechenzeit zu bewältigen. Allerdings divergieren Anspruch und tatsächliche Leistungsfähigkeit der Systeme nicht zuletzt im Bereich der Syntaxverarbeitung in beträchtlichem Masse. Standard-Parsing-Verfahren, die auf deklarativen, *constraint-basierten* Grammatiken beruhen (z.B. *DCG-Parser*), werden häufig sequentiell abgearbeitet; dies wirkt sich bei linguistischen Phänomenen wie Variationen in der

[4] Vgl. NEGRA-corpus 1998.
[5] Beteiligt sind unter anderem Computerlinguisten, Informatiker, Psychologen und Philosophen.

Wortfolge[6], diskontinuierlichen Konstituenten[7] und strukturellen Ambiguitäten, die nur mit einem parallelen Verarbeitungsansatz effizient verarbeitet werden können, nachteilig auf die Leistungsfähigkeit des Systems aus.[8]
Dem Echtzeit-Anspruch wirkt zudem der Umstand entgegen, dass herkömmliche Verfahren, wie bereits erwähnt, auf Informationen aufbauen, die die Sprachkompetenz eines idealen Sprechers und nicht den Sprachgebrauch in realer schriftlicher oder mündlicher Kommunikation kodieren. Dies gefährdet zwar nicht prinzipiell die effiziente Verarbeitung, wohl aber die angemessene Verarbeitung überhaupt: Schliesslich möchten wir möglichst adäquate und wahrscheinliche Analysen aller zu verarbeitenden Sätze, nicht jedoch sämtliche Analysevarianten derjenigen Sätze, die korrekt, d.h. gemäss der Grammatik wohlgeformt sind.
NEGRA („Nebenläufige Grammatische Verarbeitung") strebt eine Integration von Kompetenz- und Performanzdaten an, um einerseits in der Lage zu sein, theoretisch fundierte linguistische Analysen vorzunehmen, und um andererseits reale Sprachdaten beiziehen zu können, wenn es darum geht, aus bereits vorhandenem Material Informationen zur Analyse weiterer Daten zu extrahieren.

1.2.3 Das linguistische Korpus

Der performanzlinguistische Ansatz von *NEGRA* (und ähnlicher Projekte) liegt darin, dass eine Sammlung von realen Sprachdaten zusammengestellt und mit linguistischen Interpretationen verbunden wird und dass die qualifizierten empirischen Daten, ein sogenanntes *Korpus* darstellend, mit statistischen Methoden ausgewertet werden.
Die erzeugten Korpora bieten „adäquate Beschreibungen tagtäglich auftretender Sprachphänomene sowie Angaben über deren Häufigkeit"[9] – so brachte *NEGRA* z.B. das erste linguistisch interpretierte deutsche Textkorpus hervor, das aus ca. 20'000 Sätzen (Stand März 1999) Zeitungstext der Frankfurter Rundschau besteht und sukzessive erweitert wird.

Aus der Perspektive der *Stochastik* stellt das *Korpus einer Sprache*, d.h. die Gesamtheit aller sprachlichen Phänomene zu einem festgelegten Zeitpunkt, die *statistische Grundgesamtheit* dar. Bei den meisten empirischen Untersuchungen ist es aufgrund zeitlicher oder allgemein methodischer Einschränkungen nicht möglich, die Grundgesamtheit zum Gegenstand zu machen – man muss folglich eine Auswahl treffen. Um trotzdem repräsentative und unverzerrte

[6] Man denke an sogenannte „free word order languages" wie das Deutsche, wo syntaktische Informationen in erster Linie morphologisch als Flexionsendungen kodiert werden und deshalb nur in geringem Ausmasse an eine feste Position im Satz gekoppelt sind.
[7] Unter diese Kategorie fällt z.B. das Problem von finitem Verb und abgetrenntem Verbpräfix.
[8] Vgl. Project C3.
[9] Vgl. NEGRA-corpus 1998.

Ergebnisse zu erhalten, stellt man eine Zufallsstichprobe aus der Grundgesamtheit zusammen; in der Linguistik würde man hier von einem *Auswahlkorpus* sprechen.[10]

2 Darstellung syntaktischer Informationen in NEGRA

Das *NEGRA-Korpus* beinhaltet neben rohen Sprachdaten die zugehörigen syntaktischen Informationen. Man spricht, einen Bezug zu den „Syntaxbäumen" herstellend, von einer sogenannten „treebank".[11] In diesem Kapitel geht es darum, aufzuzeigen, wie syntaktische Informationen in *NEGRA* kodiert sind und inwiefern sich diese Kodierung sowohl von Syntaxannotationen anderer *Treebanks* als auch von Syntaxannotationen, wie sie aus gängigen Kompetenz-orientierten Grammatiken hervorgegangen sind, unterscheiden. Ich werde mich auf die Behandlung der wichtigsten Konzepte beschränken, da der zugrunde liegende Gegenstand in der Seminararbeit von Björn Metzinger[12] eingehender behandelt wird.

2.1 Prinzipien der Treebank-Annotation

Existierende Schemata zur Treebank-Annotation weisen einen ähnlichen Aufbau auf, was sich darauf zurückführen lässt, dass die grundsätzlichen Anforderungen an eine syntaktische Annotation feststehen und dass sich entsprechende Projekte zumeist auf den angelsächsischen Sprachraum beziehen.[13]

- Das *Prinzip der „descriptivity"*

Dieses Prinzip besagt, dass grammatische Phänomene nicht erklärt, sondern bloss beschrieben werden sollten. Schliesslich stellt sich bei jeder wissenschaftlich fundierten Erklärung das Problem der Festlegung auf ein Modell, der Theorieabhängigkeit und der Einengung des Gegenstandes, was eine Wiederverwertbarkeit der Ergebnisse erschwert.

- Das *Prinzip der „theory-independence"*

[10] Vgl. Sachs 1976, 8ff.
[11] Z.B. Skut et al. 1997[2], 1.
[12] Metzinger 2000.
[13] Vgl. Skut et al. 1997[2], 1.

Die Syntaxannotation sollte nicht auf den Erkenntnissen einer bestimmten Grammatiktheorie beruhen, wohl aber genug Daten zur Verfügung stellen, um daraus theoriespezifische Repräsentationen ableiten zu können.

- Das *Prinzip der „data-drivenness"*

Mit den Repräsentationsmitteln des Annotations-Schemas müssen sich alle linguistischen Phänomene, die im Auswahlkorpus auftreten, erfassen und beschreiben lassen. Disambiguierung ist Teil der manuellen Bearbeitung des Korpus und beruht somit auf menschlicher Sprachkompetenz.

Das *Prinzip der „multi-stratal representation"*

Um den Zugriff auf die Daten und die Übersichtlichkeit der Daten zu optimieren, sollte eine Trennung von verschiedenen Beschreibungsebenen vorgenommen werden.
Entsprechend basiert die in einer typischen *Treebank* kodierte syntaktische Information auf einem kontextfreien Konstituentengerüst, das nicht-lokale Abhängigkeiten in Form von sogenannten „trace-fillers"[14] repräsentiert.

Die folgende Abbildung[15] (1) zeigt, wie die syntaktische Struktur eines deutschen Satzes in *HPSG*-typischer Art relativ flach dargestellt ist und wie durch Platzhalter und Pfadangaben eine Beziehung zwischen verschobenen Konstituenten und ihrer „ursprünglichen"[16] Position im Satz hergestellt wird. Syntaktische Einheiten und Relationen werden durch die Phrasenkategorien in den Knoten sowie durch die Kanten hervorgehoben.

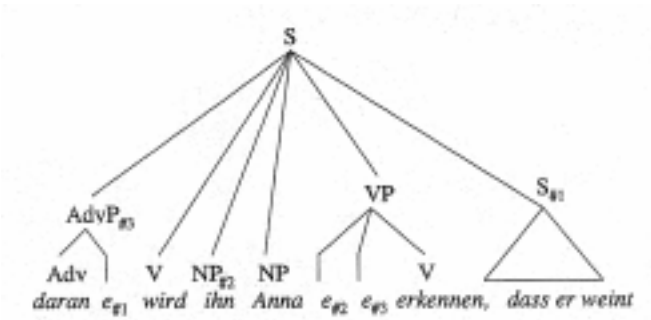


Abbildung 1: Die Trace-filler-Repräsentation

[14] *Trace-fillers* sind nichts anderes als Platzhalter an leeren Konstituentenpositionen mit Angaben der neuen Position, die die verschobene Konstituente bzw. der verschobene Teil einer Konstituente einnimmt.
[15] Vgl. Skut et al. 1997[2].
[16] Diese „ursprüngliche" Position wird von der entsprechenden Grammatik- bzw. Syntaxtheorie bestimmt.

2.2 Annotationsschemata für Free Word Order Languages

Die oben angesprochenen Prinzipien lassen die Dominanz der auf der *generativen Transformationsgrammatik* beruhenden Syntaxmodelle erkennen. Mögen diese Modelle für das Englische angemessene Werkzeuge liefern, so ist deren Anwendung auf Sprachen mit relativ freier Wortfolge wie das Deutsche problematisch: Lokale und nicht-lokale Abhängigkeiten sind weniger klar abgrenzbaren Klassen von Phänomenen als vielmehr einem Kontinuum mit fliessenden Übergängen zuzuordnen, es existiert eine Vielzahl von diskontinuierlichen Konstituententypen[17], und die Grammatikalität von verschiedenen Wortfolgen lässt sich durch die Wertzuweisung „wohlgeformt" bzw. „nicht wohlgeformt" nicht adäquat ausdrücken, da morphosyntaktische Informationen nicht in der Konstituentenposition, sondern vielmehr in der Konstituente selbst[18] als Flexionsmerkmale kodiert sind und sich dadurch für die Anordnung der Konstituenten keine zwingende Reihenfolge aufdrängt.
Vor allem der Aspekt der Auftauchenshäufigkeit diskontinuierlicher Konstituententypen ist ausschlaggebend für die Bemühungen, einen Beschreibungsformalismus für Syntaxstrukturen zu schaffen, der nicht auf dem *Trace-filler*-Mechanismus basiert und somit wesentlich transparenter und vom Menschen leichter zu handhaben ist.

NEGRA versucht, den erwähnten Ansprüchen an die Kodierung syntaktischer Informationen innerhalb eines Korpus durch die Verwendung der *Argumentstruktur* („argument structure") gerecht zu werden. Der *Argumentstruktur*-Formalismus lässt tendenziell einfache und flache Baumstrukturen zu, die insofern ungeordnet sind, als Äste sich überkreuzen dürfen, und die einen grossen Teil der Information nicht in den Knoten der syntaktischen Phrasen, sondern in funktionalen Beschreibungen der Äste („function labels", *grammatische Funktionen* bzw. „grammatical functions" repräsentierend) speichern. Das Prinzip der überkreuzenden Äste ermöglicht eine einheitliche Darstellung sowohl von lokalen als auch von nicht-lokalen Abhängigkeiten, und das Prinzip der „function labels" bietet den Vorteil, dass pro Satz eine einzige Struktur zur Darstellung theorieunabhängiger und zur Ableitung theoriespezifischer syntaktischer Information genügt.

Dem Satz von Abbildung 1 wird in Abbildung 2 eine Argumentstruktur zugewiesen, wobei in der vereinfachten Darstellung weder Part-of-speech-Tags noch „Labels" grammatischer Funktionen auf den Kanten, sondern nur phrasale Kategorien in den Knoten berücksichtigt sind[19]:

[17] Vgl. Skut et al. 1997[2]: z.B. Topikalisierung, Extraposition, getrennte NPs und PPs usw.
[18] Eigentlich ist die Wortform Träger der morphosyntaktischen Informationen. Je nach Abstraktionsgrad und Subkategorisierungsformalismus erscheinen diese Informationen jedoch auch auf der Ebene der Konstituenten.
[19] Vgl. Metzinger 2000.

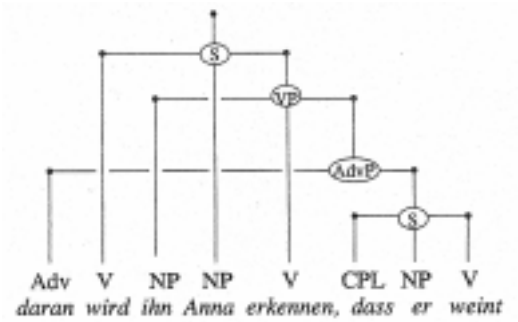


Abbildung 2: Die Argumentstruktur

Zusammenfassend kann gesagt werden, dass das *NEGRA-Korpus* kontextfreie Strukturen mit möglicherweise kreuzenden Kanten enthält. Letztere lassen sich automatisch in „traces" transformieren, falls man ein Korpus-Format, das identisch mit demjenigen der im englischsprachigen Raum häufig verwendeten *Penn-Treebank* ist, favorisiert oder benötigt.[20]

3 Gewinnung syntaktischer Informationen unter Verwendung stochastischer Methoden

„It must be recognized that the notion of a ‚probability of a sentence' is an entirely useless one, under any interpretation of this term." (N.Chomsky[21])

Chomskys wissenschaftliche Autorität hielt denn auch statistische Konzepte für längere Zeit von der traditionellen Linguistik fern. Erst in der zweiten Hälfte der 1980er Jahre verhalfen Arbeiten über *Wortarten-Tagging* („Part-of-speech-tagging", „POS-Tagging") von Church (1988), DeRose (1988) und Garside (1987) der Idee zum Durchbruch, dass es möglich sei, mit Hilfe statistischer[22] Methoden die POS-Disambiguierung aus dem bis anhin als monolithisch erachteten Problem der

[20] Vgl. NEGRA-corpus 1998.
[21] Nachzulesen in: Chomsky, N. (1969). Quine's Empirical Assumptions. In: D. Davidson and J. Hintikka (Hgs.). Words and Objections. Essay on the Work of W. V. Quine. Dordrecht.
[22] In der vorliegenden Arbeit verwende ich die Adjektive „statistisch" und „stochastisch" mehr oder weniger synonym. Genau genommen behandelt die *Stochastik* „alles auf der Wahrscheinlichkeitsrechnung Basierende" (Sachs 1976, 1), während die *Statistik* es sich vor allem zur Aufgabe macht, Ergebnisse einer Messung oder Zählung darzustellen und zusammenzufassen (vgl. ebd.).

maschinellen Verarbeitung natürlicher Sprache herauszulösen.[23] Statistische Methoden, so die Erkenntnis, liefern zwar nicht eine exakte Lösung der Probleme, rücken jedoch vernünftige Approximierungen in erreichbare Nähe.

3.1 Überblick über die Module von NEGRA

NEGRA, wie es sich im Herbst 1998 präsentiert, besteht im Wesentlichen aus vier Modulen.
- Im Hintergrund stellt eine *SQL-Datenbank*, die das *NEGRA-Korpus* repräsentiert, einerseits Sprachdaten und andererseits passende syntaktische Informationen bereit.
- Das Annotationstool *@nnotate*[24] ist die eigentliche „Werkbank" des Linguisten: Die graphische Benutzeroberfläche ermöglicht eine primär manuelle, aber effiziente strukturelle Annotation, die je nach Automatisierungsgrad mehr oder weniger von einer Anbindung an die im Hintergrund laufenden *Tagger* bzw. *Parser* profitiert.
- Die interaktive Annotation wird unter anderem vom *statistischen Part-of-Speech-Tagger TnT* unterstützt, der nicht nur den einzelnen Tokens *POS-Tags* zuweist, sondern auch grammatische Funktionen und phrasale Einheiten bestimmt und ihnen entsprechende „function labels" und „phrasal categories" zuordnet.
- Am Rande erwähnt sei hier noch der NP-Chunker *Chunkie*, der mit äusserst geringer Fehlerrate sowohl einfache als auch komplexe NPs und PPs erkennt. Er leistet die Vorarbeit, von der dann der *TnT* profitiert.

3.2 Statistisches Tagging in NEGRA

In der Folge soll das Hauptaugenmerk auf die statistische Syntaxanalyse gelegt werden: Zum einen werden grundsätzliche Überlegungen zur theoretischen Basis angestellt, zum anderen werden, nicht zuletzt in Bezug auf *NEGRA*s *TnT*-Tagger, Möglichkeiten der konkreten Verwendung und Implementation der entwickelten Modelle aufgezeigt.

---

[23] Vgl. Young and Bloothooft 1997, 118 ff.: Man ging davon aus, dass z.B. Part-of-speech-Disambiguierung nur durch eine umfassende Analyse von Syntax und Diskurs und unter Miteinbezug von Weltwissen bewältigt werden könne. So müsste man, um das Token „help" in den Phrasen „give John help[POS:noun]" versus „let John help[POS:verb]" korrekt zu bestimmen, die Sätze parsen bzw. die unterschiedlichen Subkategorisierungsrahmen von „give" und von „let" referenzieren. Für eine Disambiguierung von „off" in „I turned off[POS:Präposition] highway 66" versus „I turned off[POS:Verbale_Partikel] the radio" muss man „wissen", ob es sich bei der folgenden NP auf der semantischen Ebene um eine Strassenangabe oder um eine Geräteangabe handelt.
[24] Vgl. Metzinger 2000.

3.2.1 Der TnT-Tagger

TnT, ausgeschrieben „Trigrams'n'Tags" (und in der Folge ohne Anführungszeichen referenziert), kann zur Verarbeitung eines beliebigen Korpus einer beinahe beliebigen Sprache[25] verwendet werden. Ein Sprachmodell[26] für das Deutsche[27] sowie eines für das Englische[28] werden mit *TnT* mitgeliefert; weitere lassen sich generieren, indem ein Sprachkorpus gemäss dem vordefinierten Datenformat von *TnT* (siehe Abbildung 3) mit *POS-(„part of speech"-)Tags* aus einer festgelegten Menge, aus einem sog. *Tagset*[29], versehen und das „getaggte" Korpus von der Trainingskomponente verarbeitet wird.

Abbildung 3 zeigt eine Tabelle, deren linke Hälfte aus natürlichsprachlichem Eingabesatz und der (manuell oder maschinell) ermittelten bzw. wegen ihrer Wahrscheinlichkeit bevorzugten Wortart und deren rechte Hälfte aus Detailangaben über die Wahrscheinlichkeiten aller in Frage kommenden Wortarten besteht.
Tokens, die das Lexikon nicht kennt, sind mit einem Stern („*") markiert.[30]

```
                Basic          Optional
Input           Output         Extended Output
----------------+--------+-----------------------------------------
Der             ART      | ART    1.000000e+00
Mandolinen-Club NN     * | NN     1.000000e+00  *
Falkenstein     NE     * | NE     8.001280e-01  NN 1.998720e-01 *
und             KON      | KON    1.000000e+00
der             ART      | ART    1.000000e+00
Frauenchor      NN     * | NN     9.828203e-01  NE 1.717975e-02 *
aus             APPR     | APPR   1.000000e+00
dem             ART      | ART    1.000000e+00
```

[25] *TnT* kann für alle Sprachen trainiert werden, die Tokens mit einer Leerstelle („<space>") trennen. Eine weitere Restriktion betrifft die *Tagsets*: Sie müssen in *ASCII* repräsentierbar sein.
[26] Weiter unten wird ausgeführt, wie man sich ein derartiges (statistisches) Sprachmodell ungefähr vorzustellen hat. Für den Moment genügt die Assoziation mit einer (aus kontextfreien Regeln bestehenden) Grammatik, die letztendlich ebenfalls ein (deskriptives) „Sprachmodell" darstellt.
[27] Trainiert am Saarbrücker *German Newspaper Corpus* unter Verwendung des *Stuttgart-Tübingen-Tagsets* („STTS").
[28] Trainiert am *Susanne Corpus*.
[29] Ein *Tagset* umfasst also die Menge aller Tags, die von einem Tagger vergeben werden. So verarbeitet der *TnT* drei *Tagsets* auf verschiedenen linguistischen Ebenen: die *POS-Tags* auf Wortebene, die Phrasen-Tags auf Phrasenebene („phrasal categories") und die Kanten-Tags („edge labels") auf der Ebene der *NEGRA*-typischen grammatischen Funktionen („grammatical functions"). Vgl. z.B. Brants et al. 1997.
[30] Vgl. Brants 1998.

```
sächsischen      ADJA    | ADJA   1.000000e+00
Königstein       NN      | NN     7.762892e-01  NE 2.237108e-01
gestalten        VVINF   | VVINF  1.000000e+00
die              ART     | ART    9.796126e-01  PRELS 1.443545e-02 […]
Feier            NN      | NN     1.000000e+00
gemeinsam        ADJD    | ADJD   1.000000e+00
.                $.      | $.     1.000000e+00
```

Abbildung 3: Das Datenformat von TnT

Der *TnT* zeichnet sich durch *Robustheit* aus, was darauf zurückzuführen ist, dass das System Methoden zum Umgang mit unbekannten Wörtern auf mehreren Abstraktionsebenen kennt. Falls der Tagger eine Wortform nicht in seinem Lexikon finden kann, versucht er, zuerst auf der morphologischen Ebene durch Suffix-Abtrennung und –Bestimmung und später auf der Ebene der Phrasen bzw. Konstituenten durch Berechnung des wahrscheinlichsten Tags unter Berücksichtigung der angrenzenden Wortformen respektive deren *POS-Tags* zu einer adäquaten Analyse zu gelangen.

3.2.2 Zahlen und Fakten

Abgesehen von der Robustheit, der Anpassbarkeit und der Wiederverwertbarkeit sollte sich ein Tagger vor allem in den Kriterien *Effizienz* („efficiency") und *Genauigkeit* („accuracy") auszeichnen[31] - er sollte schnell sein und eine geringe Fehlerrate (100% - Genauigkeit) aufweisen. Grundsätzlich hängt die Tagging-Geschwindigkeit von der durchschnittlichen Ambiguitätsrate der bekannten Wörter und von der relativen Anzahl unbekannter Wörter (in Prozent von der Anzahl aller Wörter) eines Textes ab. Eine Konfiguration von *TnT* unter *Linux* auf einem *Pentium*-Rechner mit 500 Mhz vermag zwischen 20'000 und 50'000 Tokens pro Sekunde zu taggen.

Die folgende Tabelle (Abbildung 4) stellt Genauigkeits-Werte von *TnT* dar, die man für verschiedene Korpora und deren Standard-Tagsets ermittelt hat. In jedem von insgesamt zehn Messdurchgängen ist das jeweilige Korpus in ein neues *Trainings-Set* (90%) und *Test-Set* (10%) aufgeteilt worden, so dass deren Schnittmenge die leere Menge darstellt.[32]

Korpus	**Sprache**	**Textsorte**	**Grösse (Tokens)**	**Genauigkeit (Durchschnitt)**
NEGRA-corpus	Deutsch	Newspaper	350,000	96.7%

[31] Vgl. Volk 1999.
[32] Vgl. Brants 1998.

Penn-Treebank	Englisch	Newspaper	1,200,000	96.7%
Susanne-Corpus	Englisch	Mixed	150,000	94.5%

Abbildung 4: Werte für die Genauigkeit von TnT

Die niedrige durchschnittliche Genauigkeit beim Tagging des *Susanne Corpus* hängt mit der geringen Grösse des Korpus und dem relativ umfangreichen Tagset (ungefähr 160 Tags exklusive einiger „multi-token-tags" – das andere Extrem stellen das Tagset der *Penn-Treebank* mit 36 POS-Tags und 12 Interpunktions-Tags und das *Stuttgart-Tübingen-Tagset* („STTS") mit ungefähr 50 Tags dar) zusammen. Der Tagger bräuchte ein umfangreicheres *Trainingskorpus*, um so die Wahrscheinlichkeit, dass das erstellte Sprachmodell das zu bestimmende Phänomen „kennt" bzw. beschreibt, zu vergrössern.[33]

3.3.3 Stochastische Sprachmodelle

Ein *statistischer Tagger* (oder auch Parser) wie *TnT* kann aus der syntaktischen Information, die ein (manuell) vorbereitetes, mit linguistischen Beschreibungen versehenes Korpus zur Verfügung stellt, ein Sprachmodell generieren, das es ihm ermöglicht, neue, „rohe" Sprachdaten zu analysieren und so z.B. das im Korpus kodierte sprachliche Wissen zu erweitern oder aber das Ergebnis einem anderen Modul zur Weiterverarbeitung zu übergeben.
In der Folge führe ich Formalismen ein, die man dazu verwendet, um stochastische Sprachmodelle sowohl einzelner Sätze als auch ganzer Korpora in einer maschinenkonformen Art und Weise darzustellen.

3.3.3.1 Markov-Ketten

Markov-Ketten („Markov chains")[34] sind im Grunde genommen nichts anderes als *probabilistische endliche Automaten,* d.h. endliche Automaten, deren Kanten mit der Wahrscheinlichkeit, dass man den jeweiligen Übergang („arc", „Kante") zwischen zwei Zuständen („state", „Knoten") beschreitet, wenn man sich im Ausgangszustand des Übergangs befindet, versehen sind. Gemäss den Gesetzmässigkeiten der Stochastik muss die Summe aller Über-

[33] Vgl. Brants 1998.
[34] Mit „Markov-Prozess" bezeichnet man ein auf Wahrscheinlichkeitsgesetzen beruhendes Verfahren zur vollständigen Voraussage zukünftiger Zustände eines Geschehens aus dem gegenwärtigen Zustand des Geschehens, das auf den russischen Mathematiker A. A. Markov (1856-1922) zurückgeht (vgl. Bussmann 1990). Der Begriff der *Markov-Ketten* nimmt stärker Bezug auf das graphische Darstellungsformat mit Knoten und Kanten, meint dabei aber denselben Mechanismus.

gangswahrscheinlichkeiten, die die von einem Knoten ausgehenden Kanten „beschriften", eins („1") ergeben.
Markov-Ketten kann man sich ebenso wie *FSA*s als *Generatoren* oder als *Akzeptoren* vorstellen: Die Wahrscheinlichkeit, dass man in einem solchen Modell eine gegebene Zeichenkette generiert, ist gleich dem Produkt der traversierten Kanten beim *Generieren* der Zeichenkette; als *Akzeptor* weist die *Markov-Kette* der gegebenen Zeichenkette im Laufe des Abarbeitens eine Wahrscheinlichkeit zu, die gleich dem Produkt der traversierten Kanten ist.[35]

Die folgende Illustration (Abbildung 5) bezeichnet Start- und Endzustand (identisch) sowie drei weitere Zustände eines rudimentären Modells des Englischen – Kanten sind jeweils mit der Übergangswahrscheinlichkeit und dem konsumierten Zeichen beim Traversieren der Kante beschriftet.[36]

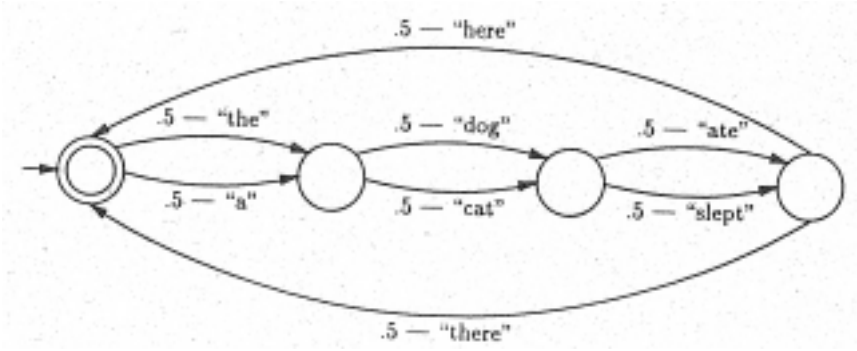


Abbildung 5: Ein einfaches Modell des Englischen

3.3.3.2 Exkurs: Kurze Einführung in die Grundlagen der Wahrscheinlichkeitstheorie

In diesem Kapitel werde ich grundlegende probabilistische Konzepte darstellen, die in den eingeführten *Markov-Modellen* verwendet werden, um dadurch einen oberflächlichen Einblick in die mathematischen Zusammenhänge zu ermöglichen.[37]

3.3.3.2.1 Ereignisse, Ausgänge und Wahrscheinlichkeiten

[35] Vgl. Charniak 1993, 32 ff.
[36] Vgl. Charniak 1993, 33.
[37] In der Folge stütze ich vor allem auf Charniak 1993, 21 ff.

Wir legen fest, dass *X* der ungewisse *Ausgang* eines *Ereignisses* sei. So könnte man sich z.B. einen Spielwürfel vorstellen, dessen einmaliger Wurf ein *Ereignis* mit ungewissem *Ausgang* darstellt – „ungewiss" insofern, als man nicht im Voraus wissen kann (einen regelmässigen Würfel vorausgesetzt), ob man eine Eins, eine Zwei, oder gar eine Sechs würfelt. Die Menge aller möglichen Ausgänge *V(X)* wäre demnach
V(X) = {1, 2, 3, 4, 5, 6}.

X wird auch eine *Zufallsvariable* („random variable") genannt. Man stelle sich z.B. vor, dass man mit geschlossenen Augen ein englisches Buch auf einer „zufälligen"[38] Seite aufschlage und mit dem Finger auf eine „zufällige" Stelle zeige. *X* wäre dann das „gefundene" englische Wort, und *V(X)* bestünde aus der Menge aller Wörter des Englischen.

Die Wahrscheinlichkeit, dass aus einer Menge von möglichen Ausgängen *V(X)* genau *X* eintrifft, nennt man *P(X = x)*, wobei dies häufig mit *P(x)* abgekürzt wird. Nun könnten wir z.B. ermitteln wollen, wie gross die Wahrscheinlichkeit ist, dass wir beim zufälligen Zeigen auf ein Wort genau das *i*-te Wort w^i des Englischen finden, d.h. wie gross $P(W = w^i)$ ist, wenn wir mit *W* alle möglichen Ausgänge des Experiments bezeichnen.

Wir nehmen an, dass es ω verschiedene englische Wörter gibt[39] und dass $| w^i |$ die Anzahl des Auftretens des *i*-ten Wortes in unserem Bücher-Korpus, der das Englische repräsentiert, bezeichnet. Entsprechend ist die Wahrscheinlichkeit, dass man das *i*-te Wort „erwischt", gleich der *relativen Häufigkeit* von w^i:

$$P(W = w^i) = \frac{| w^i |}{\sum_{j=1}^{\omega} | w^j |}$$

Gleichung 1: Die Wahrscheinlichkeit des Ausgangs eines Ereignisses (1)

Im Zähler steht, wie oft das *i*-te Wort w^i gezählt wurde, und im Nenner steht, wie viele Wörter w^j insgesamt gezählt wurden.

[38] Die *Statistik* geht mit dem Begriff der *Zufallsvariable* etwas vorsichtiger um als ich. Dort spielt es nämlich eine Rolle, ob eine zu untersuchende *Stichprobe* repräsentativ und nicht vom erhofften Ergebnis der Untersuchung beeinflusst ist, d.h. ob sie aus zufällig ermittelten Werten aus der Grundgesamtheit besteht oder ob „einfach ein paar naheliegende Werte" verwendet werden. Siehe auch Sachs 1976, 1-18.
[39] Stellen wir uns, wir hätten diese Zahl ermittelt, indem wir ein Korpus mit allen Büchern des Englischen zusammengestellt und alle verschiedenen Wörter gezählt hätten. In Wirklichkeit wird es natürlich bei einem Auswahlkorpus und annähernden Bestimmung des effektiven Werts von ω bleiben.

Interessant wird die Angelegenheit dann, wenn wir weitere Informationen über das Ereignis besitzen. Beispielsweise könnten wir zwei Wörter zufällig auswählen, so dass das erste Wort w^1 (allgemeiner: w^i) unmittelbar gefolgt wird vom zweiten Wort w^2 (allgemeiner: w^j). Wir sprechen also von zwei verschiedenen *Ereignissen* W_1 und W_2, deren Menge aller möglichen Ausgänge, wenn sie isoliert evaluiert würden, je der Menge aller englischen Wörter des Korpus entsprechen würden.

Nun spielt es jedoch eine Rolle, dass der Ausgang von W_1, das Wort w^i, gefolgt werden muss vom Ausgang von W_2, dem Wort w^j. Kennen wir z.B. w^i und möchten daraus w^j erraten, so würden wir vermutlich in allen Fällen von w^i, wo w^i ungleich dem Artikel „the" ist, auf den (sehr häufigen und deshalb sehr wahrscheinlichen) Artikel „the" setzen.
Falls w^i jedoch gleich „the" ist, dann würden wir aus linguistischen Erwägungen heraus die Möglichkeit verwerfen, dass auf einen Artikel „the" nochmals ein Artikel folgt und entsprechend für w^j auf irgendein Nomen oder Adjektiv setzen. Das Wissen um die grosse Wahrscheinlichkeit des Aufeinanderfolgens von Artikel und Nomen gewinnen wir ebenfalls aus unserem Korpus: Wir suchen alle Stellen, an denen ein Artikel, z.B. „the", erscheint, und erstellen eine Statistik bezüglich des darauffolgenden Wortes und dessen Wortart.
Wir sprechen dann von einer *bedingten Wahrscheinlichkeit* („conditional probability"), wenn der Ausgang eines Ereignisses vom Ausgang eines zweiten Ereignisses abhängt:

$$P(W_2 = w^j \mid W_1 = w^i) = \frac{|\, W_1 = w^i, W_2 = w^j \,|}{|\, W_1 = w^i \,|}$$

Gleichung 2: Die bedingte Wahrscheinlichkeit

Im Zähler steht, wie oft w^i von w^j gefolgt wird, und im Nenner steht, wie oft w^i überhaupt auftritt: Wir suchen z.B. alle Kombinationen „the[Artikel] – [Nomen]" und teilen diese Zahl durch die Anzahl der im Korpus auftretenden „the[Artikel]".

3.3.3.2.2 Das Bayes'sche Gesetz

Aus den obenstehenden Definitionen können wir eine Beziehung zwischen Wahrscheinlichkeiten, die unter anderem für das statistische NLP von grösster Bedeutung ist, ableiten – es handelt sich um das sogenannte *Bayes'sche Gesetz* („Bayes' Law"):

$$P(x \mid y) = \frac{P(x)P(y \mid x)}{P(y)}$$

Gleichung 3: Das Bayes'sche Gesetz

Um zu erklären, wozu wir dieses Gesetz verwenden können, bediene ich mich wiederum eines Beispiels. Stellen wir uns einen Arzt vor, der beim zu behandelnden Patienten ein Krankheitssymptom *s* beobachtet. Ausgehend von seiner Beobachtung soll er nun dem Patienten eine möglichst zutreffende Krankheitsdiagnose erstellen.
Der Arzt, sofern er in Stochastik bewandert ist, wird diejenige Krankheit *k* diagnostizieren, die „am wahrscheinlichsten" ist unter der Bedingung, dass Symptom *s* auftritt – er wird versuchen, *k* zu ermitteln, so dass *P(k | s)* möglichst gross ist. Allerdings wird er sich dabei schwer tun, *P(k | s)* für alle *k*s zu bestimmen. Viel leichter fällt es ihm, die *relative Häufigkeit*[40] *P(k)* irgendeiner Krankheit zu schätzen, und er sollte wissen (bzw. nachschlagen können), mit welcher Wahrscheinlichkeit eine Krankheit *k* das Symptom *s* hervorruft, was wiederum der Wahrscheinlichkeit von *s* unter der Bedingung, dass Krankheit *k* vorliegt, entspricht: *P(s | k)*.

Auch der Wert *P(s)* im Nenner soll uns keine Sorgen bereiten, da sich der Arzt seiner Sache bezüglich des Symptoms sicher ist, dieser Wert sich also nicht verändert und wir danach trachten können, in unserem fingierten Beispiel den Term im Zähler zu optimieren, um die maximale Wahrscheinlichkeit *P(k | s)* und damit die wahrscheinlichste Diagnose zu berechnen:

$$P(k \mid s) = \frac{P(k)P(s \mid k)}{P(s)}$$

Gleichung 4: Das Bayes'sche Gesetz angewendet

Im Übrigen können wir *P(s)* bestimmen, indem wir die Wahrscheinlichkeit, dass *s* eintrifft unter der Bedingung, dass *k* eingetroffen ist, berechnen (1), diese mit der Wahrscheinlichkeit von *k* multiplizieren (2) und die Resultate von Schritt (1) und (2) für alle *k*s aufsummieren:

[40] Um die *relative Häufigkeit* der Krankheit *k* zu bestimmen, „nimmt" der Arzt alle Krankheitsfälle von *k* und teilt sie durch alle Krankheitsfälle von allen Krankheiten überhaupt. Oder aber er weiss über die Verbreitung der Krankheit Bescheid: Falls man z.B. bei *k* von einer Epidemie spricht, so wird deren relative Häufigkeit innerhalb eines begrenzten Gebietes sehr hoch sein.

$$P(X = s) = \sum_{k \in V(Y)} P(X = s \mid Y = k)P(Y = k)$$

Gleichung 5: Die Wahrscheinlichkeit des Ausgangs eines Ereignisses (2)

3.3.3.2.3 Die Wahrscheinlichkeit einer Wortkette

Zurück zum NLP: Man könnte sich fragen, wie gross die Wahrscheinlichkeit einer Wortkette mit *n* Worten $w_{1,n} = (w_1, w_2, w_3, \ldots, w_n)$ ist. Es lässt sich nun ableiten[41], dass die Wahrscheinlichkeit einer Kette von Ereignisausgängen gleich dem Produkt der Wahrscheinlichkeiten der einzelnen Ausgänge unter der Bedingung, dass alle dem jeweiligen Ausgang vorhergehenden Ereignisausgänge eingetroffen sind, ist:

$$P(w_1, w_2, w_3, \ldots, w_n) = P(w_1)P(w_2 \mid w_1)\ldots P(w_n \mid w_1, \ldots, w_{n-1})$$

Gleichung 6: Die Wahrscheinlichkeit einer Kette von Ereignisausgängen

So müsste man z.B., um die Wahrscheinlichkeit der Wortkette $w_{1,3} = (\text{„}the\text{“}, \text{„}dog\text{“}, \text{„}bites\text{“})$ zu ermitteln, den folgenden Term berechnen:

$$P(\text{„the“})P(\text{„dog“} \mid \text{„the“})P(\text{„bites“} \mid \text{„the“}, \text{„dog“})$$

Abbildung 6: Die Wahrscheinlichkeit einer Wortkette (1)

Dank eines umfangreichen, statistisch bearbeiteten (englischen) Korpus und stochastischer Gesetzmässigkeiten und Methoden (siehe oben!) sind wir in der Lage, die benötigten Wahrscheinlichkeiten zu ermitteln. Im folgenden Kapitel wird anhand der sogenannten *Hidden-Markov-Modelle* („hidden Markov models“, „HMMs“) aufgezeigt, wie ein entsprechendes stochastisches Sprachmodell aussehen könnte.

[41] Auf die Ableitung wird der Leser verzichten müssen, da sie den Rahmen der Arbeit eindeutig sprengen würde.

3.3.3.3 Hidden-Markov-Modelle

Hidden-Markov-Modelle sind „ein stochastischer Prozess, der über den Markov-Ketten liegt und Sequenzen von Zuständen berechnet"[42]. Man könnte auch von einer Verallgemeinerung des Formalismus' der *Markov-Ketten* sprechen:

„HMMs are a generalization of Markov chains in which a given state may have several transitions out of it, all with the same symbol. (This is not allowed in Markov chains. There, from a given state, after a given symbol is output, only one next state is possible.)"[43]

Da mehrere von einem Zustand ausgehende Übergänge mit demselben Symbol „beschriftet" sein können, ist es nicht möglich, aufgrund der Ausgabe des Automaten zu bestimmen, welche Zustände abgearbeitet wurden; aus diesem Grunde spricht man von „*hidden* Markov models".[44]
Formal sind *HMM*s als Quadrupel $<s^1, S, W, E>$ definiert, wobei S die Menge aller Zustände, $s^1 \in S$ der Startzustand des Modells, W die Menge aller Ausgabesymbole und E die Menge aller Kanten bzw. Übergänge darstellt.[45]
Ein Übergang selbst ist als Quadrupel $<s^i, s^j, w^k, p>$ beschrieben, wobei $s^i \in S$ der Ausgangszustand der Kante, $s^j \in S$ der Endzustand der Kante, $w^k \in W$ ein Ausgabesymbol, das der Automat je nach Verwendung „akzeptiert" oder „generiert", und p die Wahrscheinlichkeit, dass der Übergang genommen wird, ist.
In einem *HMM* gibt es keine zwei Übergänge, bei denen sowohl s^i und s^j als auch w^k identisch wären – redundante Kanten sind somit ausgeschlossen.
Um sich ein Bild eines *HMM*s zu machen bzw. um abschätzen zu können, welche Sequenzen von Zuständen ein *HMM* traversieren kann, sollte man sich einen *endlichen Automaten* vorstellen, von dessen beliebigem Zustand Übergänge mit allen möglichen Ausgabesymbolen zu allen möglichen Folgezuständen ausgehen. Falls wir einen Übergang als „unpassierbar" deklarieren wollen, weisen wir ihm die Wahrscheinlichkeit null („0") zu.
Die sogenannte *Markovsche Annahme* („Markov assumption")[46] besagt, dass die einzige Grösse, die die Wahrscheinlichkeit eines Ausgabesymbols und damit eines Folgezustandes beeinflusst, der jeweilige Ausgangszustand ist:

$$P(w_n, s_{n+1} \mid w_{1,n-1}, s_{1,n}) = P(w_n, s_{n+1} \mid s_n)$$

$$= P\left(s^i \xrightarrow{w^k} s^j\right)$$

[42] Vgl. Volk 1999.
[43] Charniak 1993, 41.
[44] Charniak 1993, 44.
[45] Vgl. Charniak 1993, 43 f.
[46] Vgl. Charniak 1993, 44.

Gleichungen 7 und 8: Die Markovsche Annahme

Wie unschwer zu erkennen ist, entspricht die Wahrscheinlichkeit, dass auf die Folge von Zuständen $s_{1,n}$ und auf die Folge von Ausgabesymbolen $w_{1,n-1}$ der Zustand s_{n+1} und das Ausgabesymbol w_n folgt, der Wahrscheinlichkeit p des entsprechenden Übergangs, der Übergangswahrscheinlichkeit der Kante zwischen s_n und s_{n+1}.

3.3.3.4 HMMs und Trigram-Modelle

Das *n-gram-Modell*[47] ist nichts anderes als ein Sprachmodell, das die Einschränkung macht, dass nur die vorangehenden *n-1* Ausgabesymbole einen Einfluss auf die Wahrscheinlichkeit des darauffolgenden Ausgabesymbols haben. Abgesehen davon, dass das Modell die natürlichsprachliche Realität stark verzerrt, leistet es durch die Verminderung der Komplexität natürlichsprachlicher Probleme gute Dienste.
Für die häufigen *Trigram-Modelle* mit $n = 3$ gilt demnach folgende Vereinfachung bei der Berechnung einer Wortketten- bzw. Ausgabesymbolketten-Wahrscheinlichkeit:

$$P(w_{1,n}) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1,2})\ldots P(w_n \mid w_{1,n-1}) \text{ (n-gram)}$$

$$= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1,2})\ldots P(w_n \mid w_{n-2,n-1}) \text{ (3-gram)}$$

Gleichungen 9 und 10: Die Wahrscheinlichkeit einer Wortkette (2)

Wenn wir zwei „Pseudo-Wörter" w_{-1} und w_0 annehmen, die z.B. als Satzanfangsindikatoren fungieren könnten, so lässt sich obenstehende Formel vereinfacht ausdrücken:

$$P(w_{1,n}) = P(w_1 \mid w_{-1,0})P(w_2 \mid w_{0,1})P(w_3 \mid w_{1,2})\ldots P(w_n \mid w_{n-2,n-1})$$

$$= \prod_{i=1}^{n} P(w_i \mid w_{i-2,i-1})$$

Gleichungen 11 und 12: Die Wahrscheinlichkeit einer Wortkette (3)

[47] Vgl. Charniak 1993, 39 ff.

Die Wahrscheinlichkeit einer Wortkette der Länge *n* wäre demnach gleich dem Produkt der Wahrscheinlichkeiten von jedem Wort an der Stelle *i* unter der Bedingung, dass die jeweiligen zwei vorangehenden Wörter ebenfalls „vorhanden" sind. Beim statistischen POS-Tagging spricht man häufig von einem „n-gram-Fenster", welches denjenigen Ausschnitt der Wortkette bezeichnet, den man zur Analyse eines Wortes mit einbezieht.
Die folgende Abbildung (7) soll das *Trigram-Modell* der obenstehenden Gleichung in Form einer *Markov-Kette* visualisieren, wobei die Menge der Ausgabesymbole aus den Elementen *a* und *b* besteht und jede Kante einen Übergang repräsentiert, der sowohl mit einem Ausgabesymbol als auch mit der Übergangswahrscheinlichkeit versehen ist.
Es handelt sich um eine *Markov-Kette zweiter Ordnung* („second order Markov chain"), was bedeutet, dass man den aktuellen Zustand des Automaten zwar nicht aus dem letzten, wohl aber aus den zwei letzten Ausgabesymbolen herleiten kann.[48]

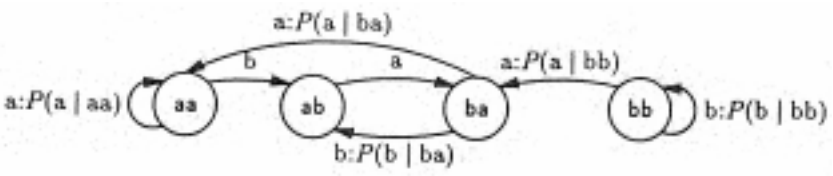


Abbildung 7: Eine Markov-Kette für Trigramme

3.3.3.5 Generierung eines Trigram-Modells

Wie bereits bei der Vorstellung des *TnT* erwähnt, benötigen wir ein *Trainingskorpus*, um ein statistisches *Trigram-Modell* zu generieren. Dieses, allgemein zwischen 80 bis 90% des gesamten Sprachkorpus repräsentierend, untersuchen wir darauf, welche Wort-Paare und Wort-Tripel mit welcher (absoluten) Häufigkeit auftreten.[49] Falls unser *Trainingskorpus* z.B. aus dem einzigen Satz „the dog bites"[50], d.h. aus der normalisierten Wortkette $w_{1\text{-}3}$ = *(„the", „dog", „bites")* besteht, so ergibt dies folgende Paare und Tripel: („the", „dog"), („dog", „bites") und („the", „dog", „bites"). (Der Übersichtlichkeit wegen habe ich davon abgesehen, die oben eingeführten Pseudo-Symbole mit einzubeziehen.)

[48] Vgl. Charniak 1993, 41.
[49] Natürlich könnten wir auch von den eigentlichen „Wörtern" abstrahieren, indem wir das Trainingskorpus zuerst manuell oder maschinell taggen und anschliessend Paare und Tripel von POS-Tags zählen.
[50] In der Realität trachtet man danach, ein möglichst umfangreiches Korpus zu verwenden, da die Wahrscheinlichkeit prinzipiell sehr hoch ist, dass man beim Anwenden des *HMM*s auf neue Texte, die nicht im *Trainingskorpus* enthalten sind, *Trigramme* entdeckt, die dem *HMM* „noch nicht bekannt sind", d.h., deren Wahrscheinlichkeit gleich null ist. Das Problem liegt, vereinfacht gesagt, darin, dass jeder beliebig langen Wort-Kette, die ein unbekanntes *Trigram* beinhaltet, die Gesamtwahrscheinlichkeit null zugewiesen wird, da das Produkt von null und beliebigen anderen Faktoren immer null ergibt. Man spricht beim stochastischen NLP oft vom „sparse data problem".

In der Folge können wir die Wahrscheinlichkeiten der einzelnen *Trigramme* ermitteln, indem wir die absolute Häufigkeit der passenden Wort-Tripel durch die absolute Häufigkeit der Wort-Paare, die aus den ersten beiden Wörtern der passenden Tripel bestehen, dividieren, wobei die Funktion C(x) (für „Count") in unserem Beispiel die absolute Häufigkeit von x ermittelt:

$$P(w_i \mid w_{i-2,i-1}) = \frac{C(w_{i-2,i})}{C(w_{i-2,i-1})}$$

Gleichung 13: Die Wahrscheinlichkeit eines Trigramms

Um auf unser Beispiel zurückzukommen: Wir könnten ermitteln wollen, wie gross die Wahrscheinlichkeit ist, dass auf „the dog" das Wort „bites" folgt. *P(„bites" | „the", „dog")* wäre also gleich dem Quotient der absoluten Häufigkeit vom Tripel („the", „dog", „bites") und der absoluten Häufigkeit vom Paar („the", „dog"). Da in unserem Beispielkorpus auf „the dog" kein anderes Wort als „bites", d.h. auf jeden Fall das Wort „bites" folgt, ist die Wahrscheinlichkeit, dass auf „the dog" „bites" folgt, gleich eins („1"). Kleiner, nämlich nur noch 0.5, wäre sie, wenn in unserem Trainingskorpus zusätzlich die Wortkette („the", „dog", „barks") stünde:
P(„bites" | „the", „dog") = ½.

Der Vollständigkeit halber muss hier noch darauf hingewiesen werden, dass das vereinfachende Beispiel den Blickwinkel allzu sehr einschränken könnte: Da in einem *HMM* theoretisch auf verschiedenen Wegen eine Wortkette generiert oder akzeptiert werden kann, ist die effektive Wahrscheinlichkeit einer Wortkette gleich der Summe der einzelnen Wahrscheinlichkeiten aller die jeweilige Wortkette produzierenden Wege.

Nun haben wir die Erkenntnis gewonnen, dass wir auf relativ einfache Art und Weise (prinzipiell!) die Wahrscheinlichkeit einer beliebig langen Wort-Kette $P(w_{1,n})$ bezüglich eines *n-gram-* bzw. *Trigram-Modells* berechnen können. Auch wenn die Präzision der beschriebenen Methode stark vom Umfang und von der Zusammensetzung des Korpus abhängt, so verspricht sie dennoch brauchbare Resultate für die Generierung stochastischer Sprachmodelle.

3.3.4 Verwendung stochastischer Sprachmodelle für das POS-Tagging

Bereits weiter oben wurde darauf hingewiesen, dass „TnT" die Abkürzung für „Trigrams'n'Tags" ist. Der *NEGRA-Tagger* verarbeitet *Hidden-Markov-Modelle zweiter Ordnung*, die gemäss dem *Trigram-Modell* aufgebaut sind. Um die Funktionsweise des Taggers zu verstehen, benötigen wir Wissen über die Verwendung der (bereits generierten) *HMM*s.

Der Unterschied zwischen dem oben beschriebenen Generieren eines *HMM*s und dem *POS-Tagging* besteht darin, dass wir bei letzterem unser Sprachmodell nicht „bloss" darauf „trainieren" können, dem Testkorpus eine möglichst hohe Wahrscheinlichkeit zuzuweisen, sondern dass wir die zusätzliche Ebene der *POS-Kategorien* berücksichtigen bzw. in unser Modell integrieren müssen.
Ein häufiger Ansatz besteht darin, die *POS-Kategorien* den Zuständen des *HMM*s zuzuordnen, so dass sich der jeweilige Zustand auf die *POS-Kategorie* des darauffolgend „produzierten" Wortes bezieht[51]. Wir können entsprechend das Problem des *POS-Tagging* folgendermassen formalisieren[52]:

$$\arg\max_{t_{1,n}} P(t_{1,n} \mid w_{1,n}) = \arg\max_{t_{1,n}} \frac{P(w_{1,n}, t_{1,n})}{P(w_{1,n})}$$

$$= \arg\max_{t_{1,n}} P(w_{1,n}, t_{1,n})$$

Gleichungen 14 und 15: POS-Tagging

Der Term *arg max$_x$ f(x)* bezeichnet den Wert von *x*, der *f(x)* maximiert, d.h. „möglichst gross macht". Wir suchen also die POS-Tag-Kette $t_{1,n}$, die die Wahrscheinlichkeit $P(t_{1,n} \mid w_{1,n})$, d.h. die Wahrscheinlich von $t_{1,n}$ unter der Bedingung, dass $w_{1,n}$ „gilt" bzw. die Wortkette $w_{1,n}$ „vorliegt", maximiert. Durch Anwendung des *Bayes'schen Gesetzes* und Gleichung 6 erhalten wir den zweiten Term der Gleichung; durch eine weitere Vereinfachung bzw. durch die Berücksichtigung der Tatsache, dass wir immer von derselben Wortkette $w_{1,n}$ ausgehen und sich deshalb der Nenner des zweiten Terms nicht verändert, erhalten wir den dritten Term der Gleichung.
Hat man einmal das durch *POS-Kategorien* erweiterte Sprachmodell generiert[53] bzw. hat man das ursprüngliche Korpus erweitert um Sequenzen von Wort-POS-Paaren mit deren Wahrscheinlichkeiten[54], so erhält man die mit der gegebenen Wortkette korrespondierende Folge von Tags, indem man die Folge der Zustände findet, die das *HMM* beim Ausgeben der Wortkette traversiert.

[51] Vorsicht: Falls wir *n* Ausgabesymbole haben, besteht die Zustandskette (bis und mit dem potentiellen Endzustand) aus *n+1* Zuständen. Da jedoch kein Wort mehr übrig bleibt, um einen vom Endzustand weiterführenden Übergang zu bezeichnen, wäre es sinnlos, diesem Zustand ebenfalls eine *POS-Kategorie* zuzuweisen – entsprechend tauchen in unseren Berechnungen nur *n* Tags auf: $t_{1,n}$.
[52] Vgl. Charniak 1993, 46-48.
[53] Details dazu sind nachzulesen u.a. in Charniak 1993, 47-48.
[54] So muss der statistische Tagger z.B. auf Informationen darüber zurückgreifen können, wie oft eine Wortfolge $w_{1,n}$ insgesamt und wie oft sie mit der POS-Folge $t_{1,n}$ assoziiert auftaucht.

3.3.4.1 Der wahrscheinlichste Weg durch ein HMM

Unsere Aufgabe beim stochastischen *POS-Tagging* wird nun darin bestehen, den wahrscheinlichsten Weg durch ein *HMM*, d.h. die Zustandsfolge mit der höchsten Wahrscheinlichkeit, bei gegebener Ausgabe-Wortkette zu bestimmen.
Der *Viterbi-Algorithmus*[55], im *TnT* ebenso wie in diversen anderen statistischen Taggern implementiert, löst unser Problem auf relativ einfache Art und Weise, indem er mit der leeren Ausgabekette beginnt und sich Ausgabesymbol um Ausgabesymbol bis zur Antwort hindurcharbeitet. Zu jedem Zeitpunkt muss der Tagger nur die wahrscheinlichste Folge von Zuständen und deren Wahrscheinlichkeit berechnen, die im „aktuellen"[56] Zustand s^i endet, und dies für alle möglichen „aktuellen" Zustände.

3.3.4.2 Besonderheiten des stochastischen NEGRA-Taggers

Wie oben bereits erwähnt wurde, weist *TnT* nicht nur *POS-Tags*, sondern auch *grammatische Funktionen* („grammatical functions") und *phrasale Kategorien* („phrasal categories") zu. Im Gegensatz zu einem gewöhnlichen stochastischen POS-Tagger, der sich für die Zuordnung der Tags eines „getaggten" Korpus mit Paaren von Wörtern und entsprechenden Tags bzw. eines universalen Markov-Modells bedient, ermittelt *TnT* lexikalische[57] und kontextuelle[58] Wahrscheinlichkeiten der grammatischen Funktionen $P_Q([\ldots])$ in Abhängigkeit von der *phrasalen Kategorie* des Mutter-Knotens *Q* – jede *phrasale Kategorie (S, VP, NP, PP, usw.)* wird also durch ein eigenes Markov-Modell repräsentiert[59]. Die Kategorien der Tochter-Knoten korrespondieren mit den Ausgabesymbolen, und die grammatischen Funktionen korrespondieren mit den Zuständen des jeweiligen Markov-Modells.

Abbildung 8 illustriert den beschriebenen Sachverhalt anhand von zwei Markov-Ketten, wobei die eine das Markov-Modell für die phrasale Kategorie *S* und die andere dasjenige für die phrasale Kategorie *VP* darstellt[60].

---

[55] Vgl. Charniak 1993, 53 ff.
[56] Ein Zustand ist dann „aktuell", wenn seine *POS-Kategorie* mit dem darauffolgenden Ausgabesymbol korrespondiert. Da in einem *HMM* verschiedene Wege zum Ziel führen können, müssen alle „aktuellen" Wege und Zustände berücksichtigt werden.
[57] Die lexikalische Wahrscheinlichkeit bezieht sich auf die „vertikalen" Beziehungen zwischen Wort und *POS-Kategorie* bzw. zwischen Wort und *grammatischer Funktion.*
[58] Die kontextuelle Wahrscheinlichkeit bezieht sich auf die „horizontalen" Beziehungen zwischen mehreren *POS-Kategorien* bzw. *grammatischen Funktionen.*
[59] Vgl. Brants et al. 1997.
[60] Vgl. Brants et al. 1997.

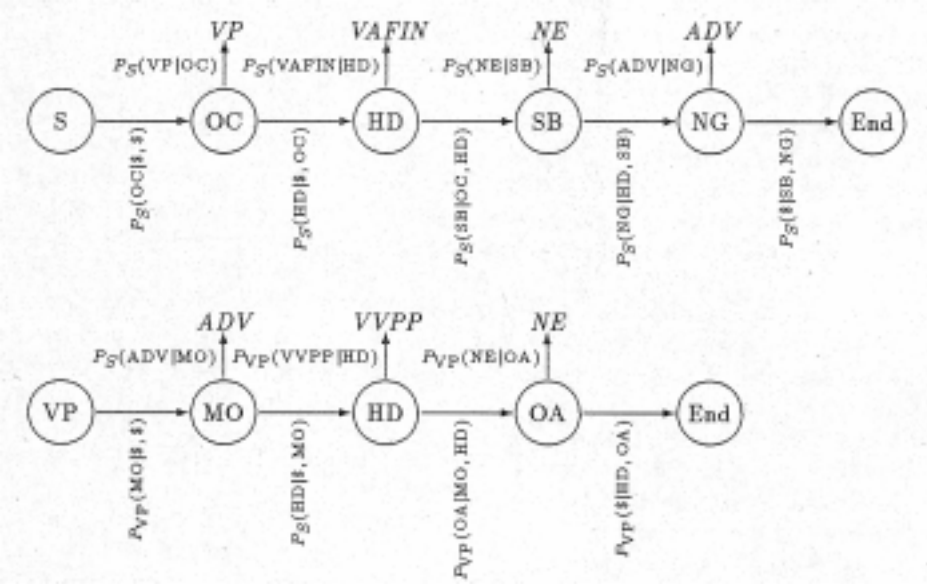


Abbildung 8 : Markov-Modelle in NEGRA

Zur Erinnerung: Bis jetzt galt, dass die Ausgabesymbole die Wörter, die Zustände hingegen die POS-Kategorien repräsentierten. Offensichtlich arbeitet der *NEGRA-Tagger TnT* auf mehreren linguistischen Abstraktionsebenen in mehr oder weniger analoger Art und Weise unter Verwendung entsprechender *Tagsets* – „Tagging" meint hier nicht bloss „POS-Tagging", sondern Syntaxanalyse im umfassenden Sinne.

4 Bibliographische Angaben

4.1 Grundlagen

[Brants 1998] Thorsten Brants. TnT – Statistical Part-of-Speech Tagging. http://www.coli.uni-sb.de/~thorsten/tnt/

[Brants and Skut 1998] Thorsten Brants and Wojciech Skut (1998). Automation of Treebank Annotation. In: Proceedings of the Conference on New Methods in Language Processing (NeMLaP-3), Sydney, Australia, January 14-17, 1998.

[Brants et al. 1997] Thorsten Brants, Wojciech Skut and Brigitte Krenn (1997). Tagging Grammatical Functions. In: Proceedings of EMNLP-2, Providence, RI, 1997.

[Bussmann 1990] Hadumond Bussmann (1990). Lexikon der Sprachwissenschaft. Stuttgart. (= 2., völlig neu bearbeitete Auflage)

[Charniak 1993] Eugene Charniak (1993). Statistical Language Learning. Massachusetts.

[Manning and Schütze 1999] Christopher D. Manning and Hinrich Schütze (1999). Foundations of Statistical Natural Language Processing. Massachusetts.

[Metzinger 2000] Björn Metzinger (2000). Syntaxannotation. Theoretische Grundlagen und praktische Anwendung am Beispiel der „Nebenläufigen Grammatischen Verarbeitung". Seminararbeit in Computerlinguistik. Zürich.

[NEGRA-corpus 1998] negr@-corpus. Projektinformationen zu NEGRA. http://www.coli.uni-sb.de/sfb378/negra-corpus/negra-corpus.html

[Project C3] Project C3: NEGRA. Nebenläufige grammatische Verarbeitung. http://www.coli.uni-sb.de/sfb378/projects/NEGRA-de.html

[Sachs 1976] Lothar Sachs (1976). Statistische Methoden. Ein Soforthelfer. Berlin, Heidelberg und New York. (=3. neubearbeitete Auflage)

[Skut et al. 1997[1]] Wojciech Skut, Thorsten Brants, Brigitte Krenn and Hans Uszkoreit (1997). Annotating Unrestricted German Text. In: Proceedings der 6. Fachtagung der Sektion Computerlinguistik der Deutschen Gesellschaft für Sprachwissenschaft, Heidelberg, Germany, 1997.

[Skut et al. 1997[2]] Wojciech Skut, Brigitte Krenn, Thorsten Brants and Hans Uszkoreit (1997). An Annotation Scheme for Free Word Order Languages. In: Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP), Washington, D.C., 1997.

[Young and Bloothooft 1997] Steve Young and Gerrit Bloothooft (Hg., 1997). Corpus-Based Methods in Language and Speech Processing. Dordrecht, Boston, London. (= Text, Speech and Language Technology Vol. 2)

[Volk 1999] Martin Volk. Vorlesung zu Morphologieanalyse und Lexikonaufbau. Teil 6. Universität Zürich. http://www.ifi.unizh.ch/CL/volk/LexMorphVorl/Lexikon06.Freq.html

4.2 Vertiefende Literatur

[Bangalore 1997] Srinivas Bangalore (1997). Complexity of Lexical Descriptions and its Relevance to Partial Parsing. Dissertation in Computer and Information Science, University of Pennsylvania.

[Brants 1997] Thorsten Brants (1997). Internal and External Tagsets in Part-of-Speech Tagging. In: Proceedings of Eurospeech 97, Rhodes, Greece, September 22-25, 1997.

[Church 1988] K. Church (1988). A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In: Proceedings of ANLP 88, Austin, Texas.

[DeRose 1988] S. DeRose (1988). Grammatical Category Disambiguation by Statistical Optimization. In: Computational Linguistics, 14(1).

[Garside 1987] R. Garside (1987). The CLAWS Word-Tagging System. In: Garside, R., F. Leech and G. Sampson (Hgs.). The Computational Analysis of English. London, New York.

[Marcus et al. 1994] Mitchell Marcus et al. (1994). The Penn Treebank. Annotating Predicate Argument Structure. In: Proceedings of the Human Language Technology Workshop, San Francisco.

[Skut and Brants 1998] Wojciech Skut and Thorsten Brants (1998). A Maximum-Entropy Partial Parser for Unrestricted Text. In: Proceedings of the Sixth Workshop on Very Large Corpora, Montréal, Québec, 1998.

[Wolfertstetter 1997] Franz Wolfertstetter (1997). Verallgemeinerte stochastische Modellierung für die automatische Spracherkennung. Aachen.