Regelbasiertes und statistisches Parsen natürlicher Sprache

Eine vergleichende Studie

Hausarbeit bei Prof. Michael Hess
Institut für Informatik der Universität Zürich
Abteilung für Computerlinguistik

Christoph Lüscher
Universitätstr. 53
8006 Zürich

Zürich, September 1999

Inhalt

1. Einführung

Dieser Text ist zu einem eigentlichen Plädoyer für statistische Parsingverfahren geworden. Nicht dass der Autor etwa die Verdienste regelbasierter (symbolischer) Verfahren bestreiten möchte, viele von ihnen stellen unter Anderem eine wichtige Basis für sogenannt „statistische", effektiv aber hybride Ansätze dar. Bei den statistischen Ansätzen handelt es sich jedoch um relative „Newcomer" im Bereich der Parsingstrategien, viele davon sind noch wenig bekannt und verdienen gerade deshalb eine besondere Beachtung (und etwas mehr Raum in diesem Text). Keine andere Entwicklung hat die Parsingtechnologie so weit vorangetrieben wie die Einführung statistischer Methoden: der „unbegrenzte" Text erscheint plötzlich in greifbarer Nähe, vormals unlösbar erscheinende Desambiguierungsprobleme fallen dahin oder werden zumindest abgeschwächt, ein falsch formulierter Satz führt nicht mehr zum Versagen der ganzen Analyseeinheit und jahrelange, mühselige Grammatikentwicklung scheint zumindest teilweise der Geschichte anzugehören. Eine solche Entwicklung kann nicht ohne Verluste ablaufen, die Abkehr vom Chomskischen „Ach, was ist die Sprache doch schwer zu beschreiben!",[1] hin zum pragmatischen „Man tut, was man kann." ist aber durchaus erfrischend.

Doch wie können die Meriten verschiedener Parsingverfahren untereinander verglichen werden? Nach wie vor gibt es keinen praktisch anwendbaren, domänenunabhängigen Parser für allgemeinen Text! Die Frage der Möglichkeit und Unmöglichkeit des Parservergleichs wird uns durch den ganzen Text hindurch beschäftigen, einen ersten Einblick gibt das folgende Kapitel.

Parsing ist der Prozess, Analysen für einen Satz zu finden, also seine Konstituenten und die Beziehungen zwischen diesen zu erschliessen. Bestimmend ist dabei die Annahme, dass sich die Bedeutung eines Satzes aus der Bedeutung seiner Komponenten und ihrer Interdependenzen erschliessen lasse. Als „Parsing" kann die morphologische, syntaktische, semantische oder gar pragmatische Analyse von Texten bezeichnet werden. In diesem Text werden wir uns hauptsächlich der Betrachtung des syntaktischen Parsing, des Prozesses, mit dem einem Satz ein Syntaxbaum zugewiesen wird, widmen. Als Hauptprobleme des syntaktischen Parsing gelten (siehe z. B. Langer, 1999, S. 11 oder Briscoe, 1996, S. 141):

- Ambiguität: Wie können syntaktische Ambiguitäten aufgelöst werden?
- Abdeckung und Robustheit: Wie können Parser entwickelt werden, die einen möglichst grossen Bereich natürlicher Sprache abdecken? Wie verhalten sich diese Parser, wenn sie auf Daten stossen, für die ihnen das zur Verfügung stehende Regelwerk keine Interpretationsmöglichkeiten anbieten? Präzision und Abdeckung schliessen sich oft gegenseitig aus: je grösser die Abdeckung eines Parsers, desto geringer seine Präzision (Pereira, 1996, S. 131)
- Effizienz: Wie können Parsingalgorithmen entwickelt werden, die mit den verfügbaren Rechenkapazitäten effizient zu verarbeiten sind?

Bevor ein Parser mit seiner Arbeit beginnen kann, muss ein Text in analysierbare Elemente zerlegt werden: eventuelle Sonderzeichen und Formatierungen müssen eliminiert, sinnvolle Word-

Tokens bestimmt werden und die zu analysierenden Sätze dürfen in der Regel keine „syntaktischen Brüche“ – die Rede ist hier zum Beispiel von Einschüben wie diesem – enthalten, Überschriften und andere speziellen Textsorten bedürfen ebenfalls einer Sonderbehandlung. Auf diesen Bereich von Problemen wird hier nur am Rande eingegangen.

Ein treffender Vergleich zwischen statistischen und regelbasierten Verfahren wird erst möglich, wenn man einen Parser als aus zwei Komponenten bestehend betrachtet:

- Einem Sprachmodell (einer Grammatik, respektive einer Sammlung von statistischen Daten und Annahmen)
- Einem Verarbeitungsalgorithmus (zu einem Satz werden eine oder mehrere Interpretationen gesucht, die am ehesten dem Sprachmodell entsprechen)

Statistische Parser unterscheiden sich von regelbasierten sehr stark im Bereich des Sprachmodells, beziehungsweise der impliziten oder expliziten Grammatik, die Verarbeitungsalgorithmen sind in manchen Fällen recht ähnlich. Statistische Verfahren erlauben aber bedeutende Leistungssteigerungen bei den Verarbeitungsalgorithmen, da sie die Datenbasis für die Anwendung fortgeschrittener Suchheuristiken zur Verfügung stellen.

Nach einer kurzen Besprechung der Möglichkeiten des Parservergleichs in Kapitel 2 gelangen im Kapitel 3 regelbasierte Parsingverfahren in unser Blickfeld, Kapitel 4 widmet sich dann etwas ausführlicher den statistischen Ansätzen.

[1] Siehe Langer, 1999, S. 29 für Erläuterungen zu dieser Frage.

2. Evaluation von Parsern

Parsing ist ein Mittel zum Zweck, die wirkliche Leistung eines Parsers könnte nur im Rahmen eines Systems zur Verarbeitung natürlicher Sprache beurteilt werden. Leider existiert bis dato kein umfangreiches System dieser Art. Alle Methoden, anhand derer die Leistungen von Parsern evaluiert werden, sind also in einem gewissen Sinn „künstlich". Nichtsdestotrotz erlauben sie einen Einblick in die Entwicklung der Kunst, wie sich gleich zeigen wird. Ein weiteres Problem beim Vergleich verschiedener Parser ist die Tatsache, dass sie alle unterschiedliche Stärken und Schwächen haben: gewisse Parser identifizieren nicht nur die Syntaxstruktur eines Satzes, sondern auch die semantischen Rollen gewisser Konstituenten, einzelne Pronomina-Referenzen etc. Wie wir später sehen werden, können statistische Parser nicht zwischen richtigen und falschen Sätzen unterscheiden. Regelbasierten Parsern gelingt dies besser, dafür haben sie meist grössere Probleme bei Sätzen mit teilweise unbekannten Komponenten.

Bis vor kurzem hat der Leistungsvergleich gemessen an der Performance über realen, unbeschränkten Sprachdaten die Entwickler von Parsern reichlich wenig interessiert. Parser wurden bestenfalls entwickelt, um beschränkte Domänen abzudecken (z. B. ein telefonisches Flug-reservationssystem), oder um computerlinguistisch interessante Phänomene (z. B. bestimmte Formen von Ambiguität) exemplarisch zu untersuchen (für eine eingehende Besprechung dieses Themas, siehe Langer, 1999, S. 29ff). Der Grund für das Desinteresse am unbeschränkten Text lag unter anderem darin, dass unbeschränkte Parser bis vor kurzem als kaum innert nützlicher Frist machbar erschienen. Erst das Aufkommen von statistischen Methoden und ihre Kombination mit bestehenden, regelbasierten Verfahren liess den unbeschränkten Text ins Zentrum des Interesses rücken.

Für regelbasierte Parser existieren kaum Daten, die einen Vergleich ihrer Leistung über unbeschränktem Text zulassen (GEPARD von Hagen Langer stellt hier eine lobenswerte Ausnahme dar. GEPARD parst 33% aller Sätze aus einem unbeschränkten Korpus, diese freilich nicht immer korrekt: Langer, 1999, S. 271ff). Erst bei den statistischen Parsern haben sich in den letzten Jahren die vergleichbaren Masse precision, recall und crossing-brackets und das Testen über unbeschränktem Sprachmaterial für den Parservergleich etabliert. Alle drei Testmasse beziehen sich auf den Vergleich eines automatisch erstellten Parses mit dem entsprechenden Parse in einer Treebank (einer Sammlung von manuell geparsten Sätzen).

Crossing-brackets: ein crossing-brackets-Fehler ist eine Konstituente im automatisch erstellten Parse, die einen Teil einer Konstituente in der Treebank-Analyse enthält, ohne diese Konstituente aber ganz einzuschliessen. Das folgende Beispiel soll dies erläutern (aus Magerman, 1994, S. 94):

```
Treebank:     [[AB][CD][EF]G]
Parse:        [[ABC] D [EFG]]
```

Die Konstituente [ABC] im Parse erzeugt einen crossing-brackets-Fehler, da sie einen Teil der Konstituente [CD] in der Treebank enthält, ohne [CD] aber ganz einzuschliessen. Die Konstituente [EFG] im Parse erzeugt hingegen keinen Fehler. Das crossing-brackets-Mass wird angegeben als der pro-Satz-Durchschnitt aller crossing-brakets-Fehler im Parse eines umfangreichen Textes.

Precision: die precision entspricht der Anzahl Konstituenten in einem Parse, die einer Konstituente in der Treebank entsprechen, dividiert durch die Gesamtzahl von Konstituenten im Parse.

Recall: entspricht der Anzahl Konstituenten in der Treebank, die einer Konstituente im Parse entsprechen, dividiert durch die Gesamtzahl von Konstituenten in der Treebank.

Exact Match: ein seltener verwendetes Mass, das des exact match, entspricht dem prozentualen Anteil von geparsten Satzstrukturen, die, inklusive Labels und Tags, exakt denen in der Treebank entsprechen.

Precision, recall, crossing-brackets und exact match-Werte werden in der Regel berechnet, indem ein grosser Teil einer Treebank (ca. 95%) als Trainingskorpus (für die Berechnung der statistischen Werte des Sprachmodells) und der Rest als Testkorpus (für das Ermitteln der Testmasse) verwendet wird. Etabliert haben sich *labelled* precision und recall-Werte: nicht nur die Struktur der Konstituenten, sondern auch ihre Labels und Tags werden verglichen.

Die beschriebenen Masse erlauben zwar einen schnellen Vergleich von verschiedenen Parsingstrategien, sie sind aber nicht über alle Zweifel erhaben, insbesondere wenn sie einzeln verwendet werden (ein Parser, der nicht parst, erreicht immer 0 crossing-brackets), oder wenn statistisch unsaubere Methoden des Testens verwendet werden: z. B. indem der Testkorpus in das Training mit einbezogen wird. Streng gesehen ist auch die Tatsache, dass Daten aus demselben Korpus für Training und Test verwendet werden, nicht statistisch korrekt. Die errechneten Werte werden insbesondere dann problematisch, wenn precision und recall hoch ausfallen: eine Treebank ist nämlich intern immer relativ inkonsistent (Magerman, 1994, S. 95, stellt fest, dass die von ihm verwendete Lancaster-Treebank nur etwa 50% interne Konsistenz aufweist, d. h. dass zwei gleiche Sätze in der Treebank nur in 50% aller Fälle exakt denselben Parse erhalten. Glücklicherweise können statistische Algorithmen über kleinere Unregelmässigkeiten in einer Treebank abstrahieren, der Testkorpus sollte aber konsistenter geparst sein, will man daraus zuverlässige Leistungsmasse ermitteln).

Auf Basis der precision, recall und crossing-brackets-Masse hat sich zwischen den drei Forschern Magerman, Charniak und Collins ein kleiner Wettlauf in Parsertechnologie entwickelt, der die Fortschritte der Entwicklung statistischer Parser gut illustriert. Die drei Forscher haben ihre Parser jeweils auf den Sektionen 2-21 der Penn Treebank (ca. eine Million Wörter) trainiert und an der Sektion 23 getestet. Den Anfang machte Magerman (1995) mit einem sehr komplexen Sprachmodell, das eine Vielzahl von Informationen einbezog, aus denen dann jeweils die relevantesten ausgewählt wurden. Dieser Parser erreichte für Sätze mit bis zu 40 Wörtern Länge eine precision von 84.9%, ein recall von 84,9% und crossing-brackets von 1.26.[2] 56.6% aller analysierten Sätze waren frei von crossing-brackets. Dem entgegen stellte Collins (1996) einen wesentlich einfacheren Parser, der auf der Berücksichtigung von Dependenzen zwischen Head-Wörtern (siehe weiter unten) in einem Parse basierte. 86.3% precision, 85.8% recall und 1.14 crossing-brackets waren das Resultat. 59.9 Sätze blieben ohne crossing-brackets. Charniak (1997-1) verbesserte dieses Resultat weiter mit einem nur unwesentlich komplexeren statistischen Modell, das syntaktisches Clustering und eine weiterentwickelte Form mathematischer Glättung verwendete und erreichte 87.4% precision, 87.5%

recall und 1.0 crossing-brackets. 62.1% aller Sätze blieben ohne crossing-brackets. Collins schlug 1997 mit 88.6% precision, 88.1% recall und 0.91 crossing-brackets zurück, indem er ein Mass syntaktischer Distanz in seine Berechnung von Dependenzen einbezog.

Diese Entwicklung ist sicherlich interessant, insbesondere wenn man bedenkt, dass die ersten statistischen Parser nur ca. 75% precision/recall erreicht haben (dies ohne Lexikalisierung, einem Verfahren, das ich im Kapitel über statistisches Parsing beschreiben werde). Dennoch stellt sich die Frage, ob die recht wenig präzisen Testmasse nicht an ihre Leistungsgrenze gelangt sind. Die Entwicklung besserer Masse oder die Integration statistischer Parser in reale Sprachverarbeitungssysteme könnten einen Ausweg aus dieser Situation darstellen. Charniak (1997-2, S. 16) empfiehlt in diesem Zusammenhang folgendes:

> „But while there is room for improvement, I believe that it is now (or will soon be) time to stop working on improving labelled precision/recall per se. There are several reasons for this. We have already noted the artificiality that can accompany these measurements, and we should always keep in mind that parsing is a means to an end, not an end in itself. At some point we should move on."

[2] Das wesentlich ernüchterndere exact match-Mass ging beim beschriebenen Wettbewerb leider verloren. Magerman (1994, S. 103ff) gibt für Tests mit der Lancaster-Treebank exact match-Werte von bis zu 37% an.

3. Abschied vom regelbasierten Parser?

Betrachtet man in Fachzeitschriften, bspw. in *Computational Linguistics*, die unaufhaltsam steigende Zahl von Artikel, die stochastischen Parsingmethoden gewidmet sind, gewinnt man den Eindruck, das hier die Hauptfront einer Entwicklung verläuft, die rein regelbasierte Verfahren immer mehr ins Hintertreffen geraten lässt.

Verschiedene Faktoren sind an dieser Entwicklung beteiligt:

1. Die bekannten Schwachstellen regelbasierter Parser (siehe hierzu zum Beispiel Tsuji, 1988, Charniak, 1994, oder Rayner und Carter, 1997). Viele dieser Schwachstellen gehen auf die Tatsache zurück, dass die Grammatiken solcher Parser Handkodierung erfordern. Die Folge:
   - Zahllose *Lücken*, eine breite Abdeckung ist meist nicht gegeben.
   - Dadurch bedingt: *mangelnde Robustheit*. Fehlt in der Grammatik eine Regel, die für das Parsen eines Satzes notwendig wäre, gelangt der Parser zu keiner Lösung. Reparaturstrategien wie Relaxation, Umschalten auf seichtes Parsen oder Inselparsen beeinträchtigen die Qualität einer Analyse unter Umständen massiv. Bedenkt man, dass eines der umfangreichsten, vorwiegend regelbasierten Systeme mit breiter Abdeckung für das Deutsche (GEPARD von Hagen Langer, bisheriger Entwicklungsaufwand: 3 Mannjahre) nur ca. 33% der Sätze aus der deutschen Tagespresse verarbeiten kann, so wird die Bedeutung dieser Einschränkungen deutlich.
   - Enormer Zeitaufwand für die Entwicklung der Grammatik
   - Geringe Portierbarkeit (zwischen verschiedenen Systemen und ganz besonders zwischen verschiedenen Sprachen)

   Eine weitere Schwachstelle ist der Umgang mit syntaktischen Ambiguitäten. Solche Ambiguitäten können sehr oft auch mit semantischen oder pragmatischen Constraints und / oder mit Lookahead-Strategien nicht aufgelöst werden. Das Auftreten mehrerer Hundert syntaktisch korrekter Parses für einen einzigen Satz scheint eher die Regel als die Ausnahme zu sein.
   Unter diesen Einschränkungen leidet die Effizienz in einem hohen Ausmass. Der Optimierung von Suchstrategien wurden zwar umfangreiche Forschungsarbeiten gewidmet, doch bleiben die Möglichkeiten unter Ausschluss statistischer und informationstheoretischer Methoden beschränkt. Man muss sich dabei vor Augen halten, dass die Optimierungsstrategien, wie sie beispielsweise ein Chartparser vom Typ des Early-Parsers verwendet, ihrerseits eine bedeutende Menge von Ressourcen in Anspruch nehmen (siehe hierzu Covington, 1994, S. 191).

2. Uszkoreit und Zaenen formulieren eine weiteres, wesentliches Problem regelbasierter Parser: „Currently no methods exist for efficient distributed grammar engineering. This constitutes a serious bottleneck in the development of language technology products.“ (1996, S. 116ff) Die Autoren drücken zwar die Hoffnung aus, dass die neue Klasse deklarativer Formalismen (constraintbasierte, bzw. Unifikations-Grammatiken) die Arbeit stark erleichtern und

beschleunigen wird, dennoch scheinen die von ihnen als bisher geltend genannten Entwicklungszyklen von 8 - 12 Jahren nach wie vor wirksam zu sein.

Langer (1999, S. 29f) weist darauf hin, „dass von den meisten Parsingsystemen überhaupt nicht bekannt ist, ob und mit welchem Erfolg sie jemals auf reale Daten angewandt wurden. In den Fällen, wo Parsingresultate publiziert wurden, handelte es sich zumeist um exemplarische Einzelanalysen für erfundene Kunstsätze, die die grundsätzliche Funktionalität des Systems, Aspekte des verwendeten Grammatikformalismus oder der Systemarchitektur illustrieren sollten, nicht aber um eine - und sei es auch noch so grobe - Evaluation der Leistung des Systems bei der Analyse von Sätzen, die von anderen Personen als den Systementwicklern produziert wurden."

Dennoch wäre ebenso voreilig wie kurzsichtig, regelbasierte Parser als „veraltet" abzutun. Eine Reihe grosser Forschungsprojekte verfolgt gegenwärtig das Ziel, domänenunabhängige, beziehungsweise portable, wiederverwendbare, elektronische Grammatiken zu entwickeln: Alvey auf GPSG-Basis in Grossbritannien (Grover et al., 1993), XTAG (auf TAG Basis) an der University of Pennsylvania, ERG (HPSG) in Stanford (CSLI), ParGram für multilinguale parallele Grammatiken auf LFG Grundlage bei Xerox, TALANA (TAG) an der Universität Paris 7 (Jussieu). Die Forschungsequippe des TALANA Projekts formuliert das Pflichtenheft einer „elektronischen Grammatik" folgendermassen:

- Abdeckung aller zentralen Sprachphänomene, die in den Grammatiken erwähnt werden
- Domänenunabhängigkeit
- Generativität (Erkennung agrammatikalischer und ambiger Strukturen)
- Informativität: unabhängig vom theoretischen Rahmen, muss eine elektronische Grammatik alle zu gleichen Syntagmen gehörigen Wörter gruppieren und ihnen eine grammatikalische Funktion zuweisen, clause boundaries erkennen und Illokutionstypen unterscheiden können.
- Parametrisierter Aufbau (Möglichkeit der Ausdehnung auf neue Phänomene bzw. der Abstraktion bereits erfasster Phänomene)
- Wiederverwendbarkeit (unabhängig vom konkreten Analyseprogramm)
- Mehrzweckfähig (Korrektur, Desambiguierung, Analyse, Generierung, Übersetzung etc.),
- Interface zu breiten lexikalischen Ressourcen.
- Interface zu phonetischen oder semantischen Ressourcen.

Bangalore und Joshi (1999) weisen zu recht darauf hin, dass die durch automatische Extraktion aus Treebanks gewonnenen Regeln „linguistisch nicht transparent und schwer modifizierbar" sind (siehe hierzu später). Dem Pflichtenheft einer „elektronischen Grammatik" werden sie somit auf keine Weise gerecht. Ihre „Robustheit" beruht u.a. darauf, dass sie ungeeignet sind, grammatikalische von agrammatikalischen Eingaben zu unterscheiden. Damit wird deutlich, dass man die Leistung regelbasierter Parsingverfahren viel weniger in Millisekunden als an der linguistischen Arbeit messen sollte, für welche sie die Konzepte und Formalismen liefern. Sie bleiben als Forschungsgrundlage unerlässlich. Insofern die aus dieser Forschungsarbeit resultierenden elektronische Grammatiken tatsächlich *unabhängig vom konkreten Analyseprogramm* und *parametrisierbar* sind, erlauben sie

zudem bei der Implementierung in konkreten Analysesystemen den Einbau effizienter statistischer Methoden, welche sowohl die Desambiguierungsleistung wie die Effizienz steigern. Ein konkretes Beispiel auf dem Gebiet der Tree Adjoining Grammars bieten Bangalore und Joshi (1999). Unifikationsbasierte Formalismen mit einem gewissen Grad stochastischer Hybridisierung bei der Implementierung konkreter Analysesysteme scheinen somit hervorragende Zukunftsaussichten zu besitzen.

4. Statistisches Parsing

4.1 Grundlagen

Ich will in diesem Kapitel stochastische Verfahren beschreiben, die zum Ziel haben, einem bestimmten Satz den (nach Möglichkeit korrekten) Syntaxbaum[3] zuzuweisen. Statistische Parser können selbstverständlich auch bei der semantischen Analyse von Sätzen behilflich sein, darauf soll aber hier aus Platzgründen nicht eingegangen werden. Hingegen arbeiten moderne statistische Parser oft mit einem Amalgam von morphologischen, syntaktischen und semantischen Methoden, um das altbekannte Problem der Desambiguierung von Satzanalysen zu lösen, und nicht selten gelingt ihnen dies auch auf recht elegante Art. Statistische Parser desambiguieren effizienter als regelbasierte Verfahren und sie sind in einem gewissen Sinne 100% robust (d. h. sie liefern zu jedem Satz eine Analyse). Wie sich aber später zeigen wird, leiden sie dafür unter anderen Problemen.

Ein statistischer Parser errechnet die Wahrscheinlichkeit eines Satzes und eines dazugehörigen Syntaxbaumes. Die „Korrektheit" eines Satzes wird somit zu einem Kontinuum. Ein Satz, respektive seine Analyse, ist mehr oder weniger „wahrscheinlich". Im Gegensatz dazu unterscheiden regelbasierte Verfahren oft zwischen korrekten und falschen (d. h. nicht analysierbaren) Sätzen (wie sich zeigen wird, hat das statistische Mass für Korrektheit viele Vorteile, aber auch einige Nachteile).

Ein statistischer Parser versucht, aus einem Korpus von handgeparsten Sätzen[4] - in diesem Kontext „Trainingskorpus" oder „Treebank" genannt – dergestalt Informationen zu extrahieren, dass er neuen, bisher nicht gesehenen Sätzen zuverlässig einen Parse zuweisen kann (die Existenz eines genügend umfangreichen Trainungskorpus soll vorläufig als vorausgesetzt gelten). Ein neu generierter Parse kann im Stil des Trainingskorpus sein, er kann aber auch zusätzliche Informationen erhalten oder eine ganz andere Form annehmen. Generell gilt es als schwieriger, eine Grammatik zu lernen, die stark von der dem Trainingskorpus zugrundeliegenden abweicht.

Ein statistischer Parser arbeitet somit in zwei Schritten: er „lernt" (das statistische Äquivalent zur manuellen Erstellung einer Grammatik) und er „parst".

Nach dem Erlernen einer „Grammatik" tut ein statistischer Parser grundsätzlich drei Dinge: a) Er sucht mögliche Parses eines Satzes, b) er schreibt diesen Parses Wahrscheinlichkeiten zu, c) er gibt den (oder die) wahrscheinlichsten Parse(s) als „korrekt" weiter. Ein statistischer Parser besteht somit aus zwei Komponenten: einem statistischen Teil, dem sogenannten *Sprachmodell*, und einem algorithmischen Teil, dem eigentlichen Verarbeitungsmechanismus. Das Sprachmodell eines statistischen Parsers ist das Äquivalent zur Grammatik eines regelbasierten Parsers und in diesem Bestandteil unterscheiden sich regelbasierte und statistische Parser besonders stark, die Verarbeitungsalgorithmen sind bei beiden Parsertypen oft ähnlich. Statistische Parser erlauben jedoch häufig effizientere Implementierungen von Parsingalgorithmen (dazu mehr später).

Formal ausgedrückt entspricht der korrekte Parse eines Satzes in der Welt eines statistischen Parsers daher der folgenden Formel:

---

[3] Respektive eine andere Beschreibung der syntaktischen Struktur des Satzes.

[4] Abenteuerlich veranlagte Computerlinguisten arbeiten auch ohne handgeparsten Korpus, allerdings meist mit geringem Erfolg. Erfolgversprechender ist die Deduktion von Grammatiken aus nur teilweise getaggten Korpora oder mittels sogenannter „Constraints", die den Lernprozess einschränken. Auf diese Verfahren werde ich später noch kurz eingehen.

$$\arg\max_T p(T \mid S, C) \qquad (1)$$

Das heisst, dass ein statistischer Parser denjenigen Parse *T* (wie Tree) sucht, der gegeben den Satz *S* und den Kontext *C* (wie Context) am wahrscheinlichsten ist. Das Problem dieses Verfahrens liegt darin, dass keine Methode existiert, um die Wahrscheinlichkeit $p(T|S,C)$ direkt zu errechnen (dazu müsste man alle Texte kennen, die existieren können). Diese Wahrscheinlichkeit kann also nur approximiert werden. In der Art, wie diese Approximation gemacht wird, liegt einer der wichtigsten Unterschiede verschiedener statistischer Parser. Eine Vereinfachung der Formel (1), die für fast alle statistischen Parser gilt, ist die folgende:

$$\arg\max_T p(T \mid S, C) = \arg\max_T p(T \mid S) \qquad (2)$$

Der weitere Kontext eines Satzes wird zwecks Vereinfachung der zu errechnenden Statistik ausser Acht gelassen. Nun entspricht $\arg\max_T p(T \mid S)$ laut dem Gesetz abhängiger Wahrscheinlichkeiten $\arg\max_T \frac{p(T,S)}{p(S)}$ und da für einen bestimmten Satz p(S) konstant bleibt, kann der beste Parse T auch gefunden werden, indem man

$$\arg\max_T p(T, S) \qquad (3)$$

berechnet. Dieser Zusammenhang ist nur insofern wichtig, als bestimmte Parser p(T|S) maximieren und andere wiederum p(T,S).

Auf welchem Weg könnte nun p(T|S), respektive p(T,S) approximiert werden?

4.2 Wortartenbestimmung

Damit ein Parser den korrekten Syntaxbaum zu einem Satz berechnen kann, muss er über Informationen betreffend der syntaktischen Kategorien der Wörter im untersuchten Satz verfügen. Die heute verfügbaren statistischen Verfahren zur Bestimmung von Wortarten, inspiriert von Methoden der Erkennung gesprochener Sprache, sind beachtenswert effizient und elegant. Aus Platzgründen kann ich hier dennoch nicht ausführlich darauf eingehen. Die Erfolge statistischer Taggingverfahren sind der Hauptgrund dafür, dass statistische Ansätze nun auch für die Analyse von Syntaxstrukturen und von menschlicher Sprache im Allgemeinen in Betracht gezogen werden.

Es gibt verschiedene Möglichkeiten, Tagging und Parsing zu verbinden: a) Dem Parsingvorgang geht ein separater Taggingvorgang voraus, an den Parser wird entweder ein eindeutig getaggter Satz, oder eine Sammlung von unterschiedlich wahrscheinlichen Worthypothesen weitergegeben. b) Der Parser bezieht alle möglichen Wortarten, d. h. das Lexikon, in seine Untersuchung mit ein. Auch Lexika lassen sich probabilistisch interpretieren, da gewisse Wortarten für bestimmte Wörter je nach Kontext mehr oder weniger wahrscheinlich sind. Im Fall a) muss der Tagger, im Fall b) der Parser

dafür sorgen, dass Sätze, die unbekannte Wörter enthalten, trotzdem möglichst korrekt geparst werden. Dieses Robustheitsproblem wird meist ebenfalls mittels statistischer Verfahren gelöst, zum Beispiel, indem die Wortart eines Wortes aus den letzten n Buchstaben seiner Endung und/oder den ersten n Buchstaben am Wortanfang geschätzt wird (ein englisches Wort, das mit *–ly* endet, ist mit hoher Wahrscheinlichkeit ein Adverb). Statistische Taggingverfahren sind sehr effizient (die besten statistischen Tagger setzen ca. 98% aller Tags korrekt – z. B. der TreeTagger von Schmid, 1995, für das Deutsche) und sie haben regelbasierte Taggingverfahren in vielen Anwendungsbereichen abgelöst. Dies geschah allerdings nicht, weil die statistischen Verfahren präziser sind als regelbasierte (letztere erreichen ca. 99.5% Genauigkeit – siehe z.B. Chanod und Tapainen, 1995, oder Samulesson und Voutilainen, 1997), sondern weil der Entwicklungsaufwand eines Regelwerkes für das regelbasierte Tagging in keinem Verhältnis zum Gewinn an Präzision steht. Im Folgenden soll der Einfachheit halber angenommen werden, dass beim Beginn des Parsingvorganges die komplette Sequenz von Wortarten-Tags zu einem Satz bekannt ist (was allerdings nicht bedeutet, dass der Parser nicht auf die einzelnen Wörter eines Satzes zugreifen darf).

4.3 Die Anfänge: Probabilistische Kontextfreie Grammatiken

Einer der simpelsten Mechanismen für die stochastische Erschliessung von Syntaxstrukturen, sind sogenannte *PCFGs*, Probabilistic Context Free Grammars (siehe z.B. Charniak, 1993, und Charniak, 1997-2).

Eine PCFG ist eine kontextfreie Grammatik, in der jeder Regel eine Wahrscheinlichkeit zugeordnet ist, nämlich die Wahrscheinlichkeit, dass eine bestimmte Konstituente mit dieser Regel expandiert wird und nicht mit einer anderen, die dafür ebenfalls in Frage käme. Ein Beispiel soll dies verdeutlichen: laut der Spielzeuggrammatik von Tabelle 1 wird in 50% aller Fälle eine *np* mit *det noun* expandiert und nur in 5% aller Fälle mit *np np*. Man beachte, dass die Wahrscheinlichkeiten aller Regeln für eine bestimmte Konstituente (z.B. für eine *np*) zusammen 1 ergeben.

s → np vp	(1.0)	np → det noun	(0.5)
vp → verb np	(0.8)	np → noun	(0.3)
vp → verb np np	(0.2)	np → det noun noun	(0.15)
		np → np np	(0.05)

Tabelle 1: Spielzeuggrammatik aus Charniak, 1997-2, S. 9

Eine umfangreichere Grammatik als die oben abgebildete kann leicht aus einer Treebank extrahiert werden, indem alle Regeln aus den handgeparsten Syntaxbäumen erschlossen und die Zahl ihrer Anwendungen gezählt wird. Die Wahrscheinlichkeit, dass eine bestimmte Regel r zur Expansion einer Konstituente c verwendet wird, kann dann geschätzt werden aus:

$$p(r(c)) = \frac{C(r,c)}{\sum_{i=1}^{n} C(r_i,c)} \qquad (4)$$

In Worten: C(r,c) ist die Häufigkeit der Anwendung der Regel r zur Expansion von c. Der Nenner des Bruches gibt an, wie oft c mit einer beliebigen Regel expandiert wird (n läuft über alle möglichen Regeln für die Expansion der Komponente c). Die Division dieser beiden Zahlen ergibt eine sogenannte *Maximum Likelihood-Schätzung* der Wahrscheinlichkeit, dass c durch r expandiert wird. Eine solche Schätzung wird um so präziser, je häufiger die geschätzten Phänomene beobachtet werden. Zum sogenannten Sparse Data-Problem und zum Problem der Verallgemeinerbarkeit solcher Schätzungen, siehe unten.

Hat man nach diesem Verfahren eine PCFG aus einer Treebank extrahiert, kann ein einfaches Sprachmodell (unser erstes!) mit folgender Formel beschrieben werden:

$$p(T,S) = \prod_{r \in T} p(r(c)) \qquad (5)$$

Das heisst: die Wahrscheinlichkeit, dass ein bestimmter Parse zusammen mit einem bestimmten Satz auftritt, entspricht dem Produkt der Anwendungswahrscheinlichkeiten aller Regeln, die zur Erstellung dieses Parses benötigt werden. Ein Parser, der auf diesem simplen Sprachmodell basiert, braucht also nichts weiteres zu tun, als z. B. mit einem chartbasierten Algorithmus alle möglichen Parses für einen bestimmten Satz zu errechnen, diesen Parses nach Formel (5) Wahrscheinlichkeiten zuzuordnen und dann den wahrscheinlichsten Parse als den „richtigen“ zurückzugeben. Dies tönt an sich einfach, doch leider stellen Programme, die diesen simplen Algorithmus implementieren, keine besonders guten statistischen Parser dar (bestenfalls erreichen sie 75% precision/recall und sie sind sehr langsam, siehe hierzu Charniak 1997-2, S. 11). Hierfür lassen sich gute Gründe finden:

- Die Formel (5) stellt ein reichlich krudes Sprachmodell dar. Insbesondere wird angenommen, dass die Wahrscheinlichkeit der Anwendung einer Regel unabhängig vom Kontext ihrer Anwendung sei (deshalb wird diese Art von Grammatiken auch als „kontextfrei“ bezeichnet). Das heisst, dass zuvor oder später angewandte Regeln (Eltern- und Kinderregeln) genausowenig berücksichtigt werden wie die einzelnen Worte, aus denen ein Satz besteht (abgesehen von ihrer syntaktischen Kategorie). Eine Annahme, die ganz klar falsch ist.
- Die Schätzungen, aufgrund derer die einzelnen Anwendungswahrscheinlichkeiten für Regeln ermittelt wurden, sind unter Umständen nicht präzise: entweder wurden solche Regeln im Trainings-Korpus zu wenig angewandt, oder ihre Anwendung im Trainingskorpus entspricht nicht der Anwendung im allgemeinen Text.
- Der Parser hat keine Mechanismen, um mit syntaktischen Konstruktionen umzugehen, die nicht in der aus dem Trainingskorpus extrahierten Grammatik enthalten sind. Ein solcher Parser scheint zwar robust, da er aus der Unzahl von vorhandenen Regeln meist dennoch für jeden Satz einen Parse ableiten kann, Parses können im Falle von fehlenden Regeln aber völlig unsinnig ausfallen.
- Ein solcher Parser kann in vielen Fällen nicht korrekt desambiguieren. So erhalten zum Beispiel im Falle von PP-Anbindungsambiguitäten mehrere Analysen dieselbe Wahrscheinlichkeit, falls sich nur der Ort der Anbindung einer PP verändert, nicht aber die Zusammensetzung der Regeln, die zur Analyse eines Satzes verwendet werden.

- Der Rechenaufwand eines solchen Parsers ist sehr hoch, da aus einer Treebank in der Regel eine sehr hohe Zahl von Regeln extrahiert werden.

4.4 Treffendere Sprachmodelle

Auf zu neuen Ufern: wie lässt sich das recht unzutreffende Sprachmodell aus Gleichung (5) verbessern, um zu einem leistungsfähigeren Parser zu gelangen?

Ein Problem bei Gleichung (5) stellen wie gesagt fehlende Regeln dar. Eine einfache Lösung würde darin bestehen, nicht fertige Regeln, sondern die statistischen „Werkzeuge" zur Erstellung solcher Regeln aus dem Trainingskorpus zu extrahieren. So könnte man zum Beispiel feststellen, dass eine Nominalphrase sehr oft mit einem Artikel, sehr selten hingegen mit einer Präposition beginnt. Auf einen Artikel folgt in einer Nominalphrase sehr selten ein weiterer Artikel, Adjektive oder Nomen sind da schon wahrscheinlicher. Die Wahrscheinlichkeit, dass eine Regel r zur Expansion einer Konstituente c verwendet wird könnte also als das Produkt der Wahrscheinlichkeit ihrer einzelnen Komponenten, gegeben jeweils die vorhergehende Komponente, betrachtet werden:

$$p(r(c)) = \prod_{t_i \in r} p(t_i \mid c, t_{i-1}) \qquad (6)$$

p(np → det n) wäre also p(det|np,NULL)*p(n|np,det). Diese Werte könnten wiederum mit Maximum Likelihood-Schätzungen ermittelt werden. Die so ermittelten Regelwahrscheinlichkeiten dürften ein robusteres Mass für die in Gleichung (5) benötigten Daten ergeben. Doch auch dieses Verfahren bringt keine merkliche Verbesserung der Performance von Parsern, die auf Gleichung (5) basieren (siehe Charniak, 1997-2, S. 12). Das Problem liegt also vermutlich im unzureichenden Sprachmodell. Eine gewaltige Verbesserung dieses Modells kann mit einer sogenannten Lexikalisierung erreicht werden, davon soll jetzt die Rede sein.

Die Geschichte der Verbesserung statistischer Parser ist zu einem grossen Teil die Geschichte der Einführung von linguistischem Wissen in die angewandten Verfahren. Lexikalisierung ist ein Beispiel dafür. Mit diesem Verfahren als Basis konnte in den letzten Jahren eine Leistungssteigerung statistischer Parser von anfänglich 75% precision/recall auf mittlerweile fast 88% erreicht werden. Ziel der Lexikalisierung ist, statistische Informationen über das Verhalten einzelner Wörter in die Bewertung eines Parses einzubeziehen. Dabei kann es sich nicht um alle Wörter eines Satzes handeln (da sonst das berüchtigte Sparse Data-Problem heftig zuschlagen würde), sondern es muss die linguistisch motivierte Schlüsselidee aufgegriffen werden, dass jede Konstituente eines Satzes einen „Head" als wichtigsten lexikalischen Bestandteil habe. Heads werden *Bottom Up* berechnet, indem zum Beispiel jeder Grammatikregel eine Funktion beigefügt wird, die angibt, welche Tochterkonstituente den Head der Mutterkonstituente darstellt. Zur Regel $s \rightarrow np\ vp$ müsste zum Beispiel notiert werden, dass der Head von s dem Head der vp entspreche. Der Head der vp in $vp \rightarrow v\ np$ wäre dann v. Lexikalisierte statistische Parser bestimmen den Head einer Konstituente meist mit relativ simplen Heuristiken (z. B. mit Tabellen), getreu dem Prinzip, dass bei der Konstruktion eines statistischen Parsers möglichst wenig handgeschriebenes Regelwerk notwendig sein soll.

Magerman (1994, S. 66) bemerkt, dass er die Tabelle, die für seinen statistischen Parser SPATTER lexikalische Heads bestimmt, in einer „knappen Stunde“ geschrieben habe. Falls zu jeder Konstituente eines Satzes ein lexikalischer Head festgelegt wird, kann ein etwas treffenderes Sprachmodell konstruiert werden, als das von Gleichung (5). Zwei Statistiken werden dabei besonders oft eingesetzt. Eine, die die Anwendung einer Regel r mit dem Head h einer Phrase verbindet: p(r|h) und eine zweite, die den Head h einer Phrase mit ihrer Kategorie t und mit dem Head m der Mutterphrase verbindet: p(h|t,m). Ein erster Verbesserungsvorschlag für die Gleichung (5) wäre also:

$$p(T,S) = \prod_{c \in T} p(h(c) \mid t(c), m(c))\, p(r(c) \mid h(c)) \qquad (7)$$

Gleichung (7) ist um einiges „weitsichtiger“ als Gleichung (5), da sie die Anwendung einzelner Regeln nicht mehr als voneinander und von den einzelnen Wörtern des Satzes unabhängig betrachtet. Mit dieser Gleichung kann es einem Parser unter Anderem gelingen, die oben erwähnten Fälle von PP-Anbindungsambiguitäten korrekt zu entscheiden, da gewisse Mutterhead-Tochterhead-Kombinationen wahrscheinlicher sind, als andere. Langer (1999, S. 104ff) verwendet in seinem hauptsächlich regelbasierten Parser ein ähnliches Verfahren zur Auflösung von PP-Anbindungsambiguitäten.

Ein Parser mit Gleichung (7) wird über den Daumen gepeilt etwa 80% precision/recall erreichen, Verfeinerungen nach dem neusten Stand der Kunst können seine Leistung bis ca. 84% verbessern (siehe unten), bessere Sprachmodelle erreichen zur Zeit wie gesagt ca. 88% precision/recall.

Die Verarbeitung von Head-Wörtern stellt nicht die einzige Möglichkeit dar, die Leistungsfähigkeit von statistischen Parsern mittels der Einbeziehung lexikalischer (und anderer, meist linguistisch motivierter) Kontextinformationen zu verbessern. So arbeitet zum Beispiel der Parser von Magerman (1994, 1995) mit einer Vielzahl von statistischen Informationen, unter denen ein Algorithmus dann die für die Lösung eines bestimmten Problems wichtigsten auswählt. Collins (1996) geht von der Feststellung aus, dass sich ein Syntaxbaum als eine Sammlung von reduzierten NPs und dazugehörigen Dependenzen interpretieren lässt und schlägt in einer Erweiterung seines Modells die Einbeziehung eines Distanzmasses für die Entscheidung von Anbindungsambiguitäten vor (Collins 1997). Jüngste Untersuchungen (Collins 1996, 1997 und Charniak 1997-1) haben gezeigt, dass komplexe Modelle (wie z. B. das von Magerman), die eine grosse Zahl von statistischen Informationen einbeziehen, nicht unbedingt leistungsfähiger sind als einfachere Modelle, die dafür auf sehr treffenden Statistiken basieren.

4.5 Problembereiche des statistischen Parsing

Aufgrund der bisherigen Darstellungen erscheinen die Errungenschaften des statistischen Parsing beeindruckend: statistische Parser lösen das Problem der Ambiguität quasi *en passant*, sie sind sehr robust, portabel (auch zwischen verschiedenen Sprachen), integrieren mit Leichtigkeit morphologische, syntaktische und semantische Desambiguierungskriterien, benötigen keine langen Entwicklungszeiten und erreichen eine Abdeckung, von der rein regelbasierte Verfahren nur träumen

können. Selbstverständlich ist auch die statistische Methode des Parsings mit gewissen Nachteilen verbunden, davon soll im Folgenden die Rede sein.

Verfügbarkeit von Trainingskorpora. Statistische Daten für die oben beschriebenen Sprachmodelle können nur aus sehr grossen Mengen manuell geparster Sätzen abgeleitet werden. Grosse Treebanks, die das effiziente Training von statistischen Parsern ermöglichen, sind bis dato aber nur in englischer Sprache verfügbar (z. B. die Penn Treebank, siehe Marcus, Santorini und Marcinkiewicz, 1993). Wenn man bedenkt, welchen Entwicklungsschub das Erscheinen der Penn Treebank bei den Verfahren für statistisches Parsing ausgelöst hat, muss dieser Zustand als sehr bedenklich eingestuft werden. Die Entwicklung von grossen Treebanks für verschiedene Sprachen sollte eine relativ hohe Priorität in der computerlinguistischen Forschung erhalten. Von den Ressourcen einer Treebank könnten nicht nur rein statistische Verfahren profitieren, sondern zum Beispiel auch regelbasierte Parser, die mit statistischer Desambiguierung arbeiten.

Sind wenig Daten für das Training von statistischen Parsern verfügbar, bieten sich Trainingsmethoden an, die mit einer Mischung von handgeparstem und ungeparstem Sprachmaterial arbeiten. Im Kern des Interesses stehen hierbei Trainingsalgorithmen, die angelehnt an die sogenannten „Markov-Modelle“ für das Tagging und die Spracherkennung die statistische Struktur einer bestehenden Grammatik mit Hilfe bestimmter Algorithmen (insbesondere Baum-Welch) an einen ungeparsten Text anpassen können. Gewisse Verfahren können sogar ganz ohne handgeparstes Material auskommen, indem dem Grammatik-Lernprozess verschiedene linguistisch motivierte „Hemmnisse“ (z. B. simple, vorgegebene Kerngrammatiken, siehe unter anderem Charniak, 1993, S. 104ff) auferlegt werden. Die Leistungen solcher Verfahren sind aber äusserst diskutabel (d.h. es gelingt damit meist nicht, eine sinnvolle Grammatik zu erschliessen).

Die Entwicklung und Modifikation einer grossen Treebank ist nicht unbedingt wesentlich weniger problematisch als die Entwicklung einer umfangreichen Grammatik. Allerdings ist die Zahl möglicher Sprachmodelle, die auf der Basis einer Treebank getestet werden können, sehr gross. Eine gute Treebank bietet statistischen Verfahren quasi ein „Bild“ einer Sprache.

Sparse Data. Das Sparse Data-Problem gilt als die Achillesferse statistischer Sprachverarbeitung: statistische Sprachmodelle erfordern eine derart hohe Zahl von Parametern, dass diese selbst aus einem sehr grossen Korpus nicht immer zuverlässig geschätzt werden können (das Gesetz von Zipf ist hier besonders unerbittlich). In den letzten Jahren haben sehr intensive Forschungsanstrengungen zu diversen attraktiven Verfahren geführt, die zumindest teilweise Lösungen für dieses Problem offerieren. Um das Sparse Data-Problem zu verstehen, können wir nochmals Gleichung (7) betrachten:

$$p(T,S) = \prod_{c \in T} p(h(c) \mid t(c), m(c))\, p(r(c) \mid h(c)) \qquad (7)$$

Wollen wir p(h|t,m) mit dem Maximum Likelihood-Verfahren schätzen, müssen wir folgende Gleichung anwenden:

$$p(h \mid t, m) = \frac{C(h,t,m)}{C(t,m)} \qquad (8)$$

Das heisst, wir müssen für alle möglichen Kombinationen einer Head-Kategorie, eines Head-Wortes und eines Mutter-Head-Wortes Daten sammeln, also für eine sehr grosse Zahl von Permutationen! Falls wir Annehmen, dass eine Sprache 100'000 mögliche Head-Wörter und 30 syntaktische Kategorien aufweist, sind dies immerhin 300‘000‘000‘000 Datensätze. Gewisse Permutationen werden dabei im Trainingskorpus zwangsläufig gar nicht oder nur sehr selten auftauchen, gewisse unserer Schätzungen werden also ungenau oder schlicht falsch ausfallen. Für diese Art von Problemen lassen sich grob vier Typen von Lösungen unterscheiden: Reduktion der Komplexität des Sprachmodells bzw. Veränderung des Sprachmodells, mathematische Glättung, Clustering und – in jüngster Zeit – sogenannte „Decomposable Models“.

Reduktion der Komplexität des Sprachmodells: gewisse Sprachmodelle erscheinen zwar sehr plausibel, ihre Parameter können aber nur sehr schwer geschätzt werden. Solche Modelle sind für die Anwendung auf reale Daten ungeeignet, sie müssen angepasst werden (man nehme zum Beispiel das plausibelste aber unberechenbarste Sprachmodell aller: $p(T|S,C)=p(T|w_{1,n},cw_{1,n})$ – Die Wahrscheinlichkeit eines Parses, gegeben alle Wörter des Kontextes und alle Wörter des Satzes).

Mathematische Glättung: Anstelle des reinen Maximum Likelihood-Verfahrens kann zum Beispiel die Wahrscheinlichkeit p(h|t,m) mit einer sogenannten „Deleted Interpolation“ geglättet werden, indem auch Statistiken einbezogen werden, die einen weniger umfangreichen Kontext berücksichtigen.

$$p(h \mid t,m) = \lambda_1 \hat{p}(h \mid t,m) + \lambda_2 \hat{p}(h \mid t) \qquad (9)$$

λ_1 und λ_2 sind Gewichtungsfaktoren, λ_1 sollte bedeutend grösser sein als λ_2, um der präziseren Schätzung p(h|t,m), falls vorhanden, mehr Gewicht einzuräumen. Die Summe von λ_1 und λ_2 muss 1 ergeben. Ist also eine bestimmte Permutation von h,t und m nicht vorhanden, kann p(h|t,m) nur aus der (hoffentlich vorhandenen) Kombination von h und t geschätzt werden. Nebst der „Deleted Interpolation“ sind auch andere mathematische Glättungsverfahren möglich (zum Beispiel „Add One“).

Clustering: Clusteringverfahren zielen allesamt darauf ab, nur diejenigen Informationen für eine Entscheidungsfindung beizuziehen, die auch zuverlässig ermittelt werden können. Sie fassen bestimmte Informationen zu Gruppen (oder „Clustern“) zusammen, versuchen dabei aber möglichst wenig Informationen zu verlieren. Die Ausgestaltung verschiedener Clusteringverfahren ist sehr unterschiedlich. Ein Vorschlag für ein einfaches, explizites Clusteringverfahren wäre der folgende (vereinfacht wiedergegeben nach Charniak, 1993, S. 136ff): man zähle für jedes Wort w alle Wörter w_i, die gleich nach dem Wort w im Trainingskorpus beobachtet wurden. So erhält jedes Wort w einen Vektor der folgenden Form: $V(w)=<|w_1|,|w_2|,...,|w_n|>$. Nun fasst man alle Wörter mit ähnlichen Vektoren zu Gruppen (Clustern) zusammen. Kann man nun zum Beispiel für die Errechnung von p(h|t,m) eine bestimmte Permutation von h,t und m in den Trainingsdaten nicht finden, sucht man nach h, t und c(m), nach c(h), t und m, oder nach c(h), t und c(m), wobei c(x) „Cluster von x“ bedeutet. Nach diesem Verfahren werden Wörter zu „Verhaltensgruppen“ zusammengefasst und somit für die Schätzung der statistischen Elemente des Sprachmodells weniger Daten benötigt.

Ueberla (1997) zeigt, dass geclusterte Modelle bezüglich Präzision mit ungeclusterten nicht nur mithalten, sondern sie zum Teil sogar übertreffen (wegen der Glättung von Sparse Data-Ausreissern).

Einen anderen Weg gehen die sogenannten Entscheidungsbäume (z. B. Magerman, 1994): Entscheidungsbäume stellen Fragen über den Kontext einer zu ermittelnden Wahrscheinlichkeitsverteilung, für p(h|t,m) also zum Beispiel „Ist die Mutterkategorie ‚vp'?", oder „Ist der Head der Konstituente ‚arbeiten'?". Aus einer Sequenz solcher Fragen ermittelt ein Entscheidungsbaum dann eine bestimmte Wahrscheinlichkeit, respektive eine Wahrscheinlichkeits-verteilung. Auch Entscheidungsbäume sammeln Daten implizit zu Clustern, sie können zum Beispiel für verschiedene Permutationen von h,t und m dieselbe Sequenz von Fragen stellen und dieselbe Wahrscheinlichkeit zurückgeben (eine leicht verständliche Einführung in die Praxis der Entscheidungsbäume am Beispiel eines Taggingproblems bietet Schmid, 1994).

Decomposable Models (Bruce und Wiebe, 1999) wurden erst vor kurzer Zeit als Mittel zur Überwindung des Sparse Data-Problems beim stochastischen Parsing entdeckt. Dieses Verfahren hat zum Ziel, ein Modell, in welchem alle Variablen mit allen übrigen zusammenhängen, schrittweise zu vereinfachen, indem Unabhängigkeitsannahmen für eine wachsende Zahl von Daten aufgestellt werden. Zwar erhöht jede Unabhängigkeitsannahme grundsätzlich die Unsicherheit („Perplexität") der Prognose, doch ist die Zunahme der Perplexität oft sehr gering im Vergleich zum Vorteil der leichteren Berechenbarkeit und zur Tatsache, dass für die Berechnung eines vereinfachten Modells wesentlich weniger statistische Daten ermittelt werden müssen.

Komplexität der impliziten oder expliziten Grammatik. Eine PCFG, die aus der Penn Treebank abgeleitet wird, enthält ca. 16000 Regeln. Dies vermittelt einen Eindruck davon, wie komplex die expliziten oder impliziten Grammatiken statistischer Parser sind. Ein Parsingvorgang mit einer solchen Grammatik und einem üblichen Chartparser erzeugt durchschnittlich 1.2 Millionen Kanten pro Satz (Charniak et al., 1998). Noch erschreckender sieht es anfangs bei Parsern aus, die ganz ohne explizite Grammatiken auskommen. Der Suchraum, den aus Treebanks abgeleitete syntaktische Informationen aufspannen, ist enorm und macht die Implementierung von effizienten Suchstrategien unerlässlich. Glücklicherweise bieten dieselben statistischen Daten, die zur Errechnung von Syntaxbäumen herangezogen werden, auch eine Basis für die Anwendung fortgeschrittener Suchheuristiken. Nebst A*-Suche und Beam-Search werden auch Verfahren wie Stack Decoding (Magerman 1994) oder chartbasierte Best First-Suche (Charniak et al., 1998) angewandt. Doch selbst mit fortgeschrittenen Suchstrategien gelingt es statistischen Parsern selten, den Suchraum komplett zu erschliessen. Für Parsingsysteme in einem zeitkritischen Umfeld müssen also verlustreiche Suchstrategien angewandt werden, mit dem Risiko, den besten Parse zu „übersehen".

Statistische Parser unterscheiden nicht zwischen richtigen und falschen Sätzen. Diesen Vorwurf der „Übergenerierung" erhebt insbesondere Langer (1999, S. 19). Da statistische Parser die „Richtigkeit" eines Satzes als Kontinuum interpretieren, eignen sie sich sehr schlecht für Aufgaben, bei denen die Unterscheidung zwischen richtigen und falschen Sätzen eine Rolle spielt, zum Beispiel für Syntax-Korrekturprogramme. Die naheliegende Lösung, zur Bestimmung der Korrektheit eines Satzes einen bestimmten „Wahrscheinlichkeits-Threshold" festzusetzen, ist nicht durchführbar, da die Wahrscheinlichkeiten verschiedener Parses eines Satzes nur relativ zueinander interpretiert werden können.

Die implizite Grammatik statistischer Parser ist nicht interpretierbar. Die „Grammatik“ eines statistischen Parsers besteht aus einer unüberblickbaren Sammlung von Daten. Aussagen über ihre Angemessenheit können allenfalls aus den Fehlern des Parsers abgeleitet werden, ihr eigentlicher Inhalt bleibt der menschlichen Interpretation aber verschlossen, punktuelle Verbesserungen werden so verunmöglicht. Ein statistisches Sprachmodell wird meist aus bestimmten (linguistischen, informationstheoretischen) Prämissen abgeleitet. Seine Wirksamkeit kann aber nur empirisch getestet werden.

Erinnern wir uns an die einleuchtende Forderung der TALANA-Gruppe, dass eine „elektronische Grammatik“ parametrisierbar sein müsse, um sich verschiedenen Verwendungszwecken und Strategien anpassen zu können. Bei stochastischen Grammatiken ist dies kaum möglich, da sie nicht von Menschen lesbar sind und da man daher auch nicht wissen kann, welche ihrer Parameter verändert werden können. Dies, zusammen mit der Tatsache, dass solche Grammatiken kein normatives Parsen ermöglichen, wie es auf einigen Gebieten wünschbar oder gar notwendig ist, macht deutlich, dass es nicht darum gehen kann, generell für stochastische und gegen regelgeleitete Systeme zu optieren oder umgekehrt, sondern dass es gilt, sorgfältig abzuwägen, was wo entwickelt werden muss und wie die beiden Ansätze kombinierbar sind. Eine hybride Entwicklungslinie wird zum Beispiel von den bereits zitierten Autoren Bangalore und Joshi (1999) verfolgt. Sie zeigen, dass mit einer „Feature-based Lexicalized Tree Adjoining Grammar“, wie sie dem XTAG und dem TALANA Projekt zugrunde liegt, ein sogenanntes „Supertagging“ möglich ist, dass den Wörtern eines Satzes jeweils all diejenigen Strukturen zuweist, in die sie eingehen können, wobei die meist zahlreichen Zuweisungen durch effiziente Desambiguierungsalgorithmen (teils regelgeleitet, teils auf n-Gramm-Statistiken basierend) reduziert und dadurch die Leistung eines nachgeschalteten Parsers stark verbessert wird. Die Autoren berichten von 97.1% korrekt geparsten Sätzen aus dem Wallstreet Journal, bei einem Trainingsset von 1'000'000 Wörtern und der Berücksichtigung der 3 statisch höchstrangierenden Supertags.

5. Schlusswort – Das Beste beider Welten

Aus diesem Text sollte hervorgegangen sein, dass statistische und regelbasierte Parsingverfahren keineswegs so unvereinbar sind, wie oft angenommen wird. Beide Ansätze haben ihre Stärken und Schwächen: statistische Verfahren lernen leichter implizite „Regeln“ (!) zur Desambiguierung und ihre „so-so, la-la“-Einschätzung der Korrektheit eines Satzes hat in *den* Bereichen Vorteile, wo ein Parser aus nicht immer ganz korrekten Eingaben das Beste machen muss. Regelbasierte Parser hingegen erkennen deutlicher die Grenzen zwischen Grammatikalität und Ungrammatikalität – und genau diese Unterscheidung benötigen statistische Parser zur Erschliessung des schier unbegrenzten Suchraumes möglicher Kombinationen einzelner Bestandteile der menschlichen Sprache. Lösungen für das berüchtigte Sparse Data-Problem und statistische Sprachmodelle im allgemeinen basieren oft auf Wissen, das bei Untersuchungen mit regelbasierten Systemen gewonnen wurde, die Idee zur Verwendung lexikalischer Heads entsprang nicht dem „Datenbrei“ eines statistischen Parsers.

Menschliche Sprache scheint beides zu sein: ein von Regeln beherrschtes System (wie es die Rationalisten, z. B. um Chomsky behaupten) und eine Ansammlung von erlernten statistischen Regularitäten, Assoziationen und Konstruktionen (wie es die unerbittlichen Statistiker behaupten). Ein System, das eine solche Sprache verarbeiten will, muss vermutlich das Beste beider Welten kombinieren.

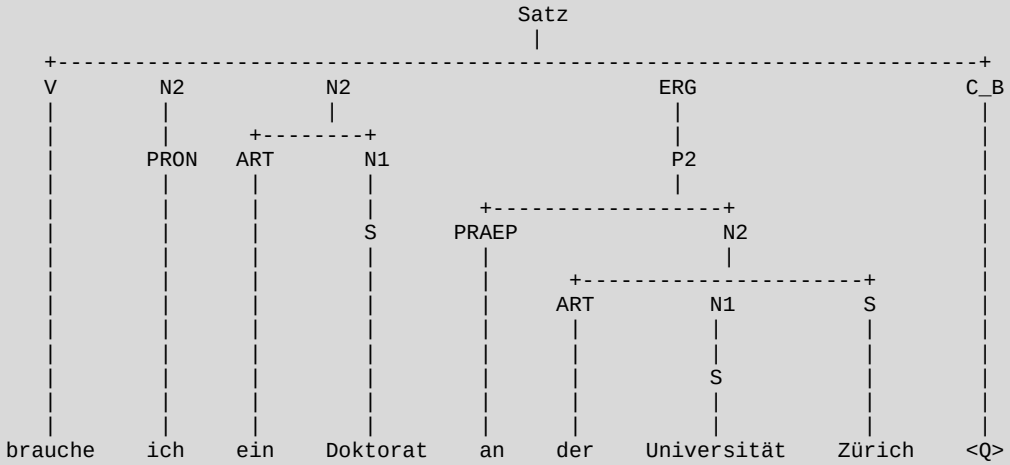
In diesem Sinne gehört das letzte Wort dem noch jungen Parser des UIS-Projektes der Computerlinguistischen Abteilung der Universität Zürich,[5] der auf die Frage „Wieviele Semester brauche ich für ein Doktorat in Computerlinguistik an der Universität Zürich?“ etwas nihilistisch reagiert:

Parsing-Ergebnis (Ausschnitt)

[Wieviele,Semester,brauche,ich,für,ein,Doktorat,in,Computerlinguistik,an,der,Universität,Zürich,<Q>]

Der Parser kann keine vollständige Lösung finden.

Arnold Buchers Repair-Modul wird aufgerufen und sucht nach Teillösungen.
Die Lösungen erhielten die Wertung: 3.0



[5] Zu testen auf: http://www.ifi.unizh.ch/cgi-bin/mvolk/UIS-Parser/uis_cgi.pl, Datum der wiedergegebenen Ausgabe: 16. September 1999.

6. Literaturliste

Allen, J. 1995. *Natural Language Understanding*. Menlo Park: Benjamin/Cummings. Kapitel 7.

Bangalore, S. und A. K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguistics* 25, 2, S. 237-265.

Briscoe, T. 1996. Robust parsing. In R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, V. Zue, Herausgeber, *Survey of the State of the Art in Human Language Technology*, S. 141-143. Internet: http://www.cse.ogi.edu/CSLU/HLTsurvey.

Bruce, R. und J. M. Wiebe. 1999. Decomposable Models in Natural Language Processing. *Computational Linguistics* 25, 2, S. 195 – 208

Chanod, J.-P und P. Tapanainen. 1995. Tagging French – Comparing a Statistical and a Constraint-Based Method. *Proceedings of EACL-95*.

Charniak, E. 1993. *Statistical Language Learning*. MIT Press, Cambridge.

Charniak, E. 1997-1. Statistical Parsing with a Context-Free Grammar and Word Statistics. *Proceedings of the Fourteenth National Conference on Artificial Intelligence*. Internet: http://www.cs.brown.edu/people/ec/.

Charniak, E. 1997-2. Statistical Techniques for Natural Language Parsing. Internet: http://www.cs.brown.edu/people/ec/.

Charniak, E., S. Goldwater und M. Johnson. 1998. Edge-Based Best-First Chart Parsing. Internet: http://www.cs.brown.edu/people/ec/.

Collins M. J. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, S. 184-191.

Collins, M. J. 1997. Three Generative Lexicalized Models for Statistical Parsing. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguists*.

Krenn, B. und C. Samulesson. 1997. *The Linguist's Guide to Statistics*. Internet: http://nlp.korea.ac.cl/~hjchung/nlp/Krenn97.ps.gz.

Magerman, D. M. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Internet: http://xxx.lanl.gov/abs/cmp-lg/9405009.

Magerman, D. M. 1995. Statistical Decision-Tree Models for Parsing. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguists*, S. 276-283.

Marcus, M. P., B. Santorini und M. A. Marcinkiewicz. 1993. Building a Large Annotated Corpus for English: the Penn Treebank. *Computational Linguistics* 19, S. 313-330

Naumann, S. und H. Langer. 1994. *Parsing: Eine Einführung in die maschinelle Analyse natürlicher Sprache*. B. G. Teubner, Stuttgart.

Pereira, F. 1996. Sentence Modeling and Parsing. In R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, V. Zue, Herausgeber, *Survey of the State of the Art in Human Language Technology*, S. 130-140. Internet: http://www.cse.ogi.edu/CSLU/HLTsurvey.

Rayner, M. und D. Carter. 1997. *Hybrid Language Processing in the Spoken Language Translator*. Internet: http://www.cam.sri.conm/tr/crc064/paper/node1.html.

Samulesson, C. und A. Voutilainen. 1997. Comparing a Linguistic and a Stochastic Tagger. *Proceedings of ACL/EACL Joint Conference*, S. 246-253.

Schmid, H. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. Internet: http://www.ims.uni-stuttgart.de/www/Tools/DecisionTreeTagger-de.html.

Schmid, H. 1995. Improvements in Part-of-Speech Tagging with an Application to German. Internet: http://www.ims.uni-stuttgart.de/www/Tools/DecisionTreeTagger-de.html.

TALANA-Projekt. Internet: http://talana.linguist.jussieu.fr/Presentation/themes.html.

Ueberla, J. P. 1997. An Extended Clustering Algorithm for Statistical Language Models. Internet: http://xxx.lanl.gov/abs/cmp-lg/94120003.

Zaenen, A. und H. Uszkoreit. 1996. Grammar Formalisms. In R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, V. Zue, Herausgeber, *Survey of the State of the Art in Human Language Technology*, S. 116-121. Internet: http://www.cse.ogi.edu/CSLU/HLTsurvey.