Stochastische lexikalisierte Baumadjunktionsgrammatiken

Ausarbeitung eines Vortrags im Proseminar
„Effizientes Parsen und Generieren mit Baumadjunktionsgrammatiken“

Oliver Plaehn
plaehn@coli.uni-sb.de

21. Januar 1998

Inhaltsverzeichnis

1 Einleitung

Seit ungefähr den achtziger Jahren werden statistische Methoden zur Verarbeitung natürlicher Sprache (*Natural Language Processing* — NLP) immer populärer, insbesondere auch aufgrund der Unzufriedenheit vieler Forscher mit den „klassischen", linguistisch motivierten Ansätzen, die bis zu diesem Zeitpunkt nur bescheidene Ergebnisse — in Hinblick auf die Leistungsfähigkeit oder überhaupt das Funktionieren real existierender Systeme — hervorgebracht haben. Nach beachtlichen Anfangserfolgen, vor allem im Bereich der Erkennung gesprochener Sprache, wurde bald klar, daß man ganz ohne klassische linguistische Methoden auch nicht auskommt, so daß man bemüht ist, diese beiden Ansätze zu kombinieren.

Diese Ausarbeitung stellt hauptsächlich einen solchen Ansatz vor, nämlich die stochastischen lexikalisierten Baumadjunktionsgrammatiken (*Stochastic Lexicalized Tree-Adjoining Grammar* — SLTAG; [Sch92]). Dazu werden zunächst in den Abschnitten 2 und 3 die zum kritischen Vergleich nötigen Grundlagen zweier älterer, gegensätzlicher Ansätze knapp dargestellt. Darauf folgt im Abschitt 4 eine ausführlichere Darstellung von SLTAG. Vergleichend wird dann im Abschnitt 5 kurz auf einen ähnlichen Formalismus eingegangen, die probabilistischen Baumadjunktionsgrammatiken (*Probabilistic Tree-Adjoining Grammar* — PTAG; [Res92]). Ein weiteres Verfahren, das *Supertagging* ([Ban97, Kapitel 4]), folgt im Abschnitt 6. Abschließend werden die verschiedenen vorgestellten Formalismen und Methoden im Abschnitt 7 kritisch bewertet und verglichen.

2 Co-occurence Statistics

Ein häufig benutztes Verfahren der statistischen Untersuchung natürlicher Sprache betrachtet Instanzen von Wörtern, die gemeinsam in einem Fenster fester Breite auftauchen. In diesem Abschnitt werden exemplarisch für diesen Bereich der *co-occurence statistics* n-gramm-Modelle erläutert, die eine Anwendung des allgemeineren Konzepts der *Hidden-Markov-Modelle* (HMM) sind. Als weitere Anwendung von HMM wird das Part-of-Speech Tagging angeführt.

Da die entsprechenden Konzepte hier nur sehr knapp ausgeführt sind, sei der Leser für weiterführende Informationen auf [Cha93] und [KS94] verwiesen.

2.1 n-gramm-Modelle

Es wird eine Sequenz von Zufallsvariablen (ZV) $W_1, W_2, \ldots, W_N$ (kurz: $W_{1,N}$) betrachtet, deren Wertebereiche jeweils alle Wörter einer bestimmten zu untersuchenden Sprache (Deutsch, Englisch, ...) umfassen. Die Indize der ZV definieren meist eine zeitliche oder räumliche Reihenfolge, z. B. die Anordnung von Wörtern in einem Satz. $P(w_i)$ steht abkürzend für $P(W_i = w_i)$ und meint die Wahrscheinlichkeit, daß die i-te Zufallsvariable der ZV-Sequenz $W_{1,N}$ den Wert w_i, also in diesem Fall ein bestimmtes Wort, annimmt.

Oft ist man an Aussagen darüber interessiert, mit welcher Wahrscheinlichkeit ein begonnener Satz mit einem bestimmten Wort fortgesetzt wird, also — wenn man die ersten i Wörter schon kennt — an $P(w_{i+1} \mid w_{1,i})$, oder an der Wahrscheinlichkeit, mit der ein bestimmter Satz produziert wird, an $P(w_{1,N})$. Die letzte Gleichung läßt sich mithilfe der ersten umformulieren zu:

$$P(w_{1,N}) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1,2}) \ldots P(w_N \mid w_{1,N-1}) \tag{1}$$

Das *n-gramm-Modell* trifft nun die Annahme, daß nur die letzten $n-1$ Wörter einen Einfluß auf die Wahrscheinlichkeit des nächsten Wortes haben. Diese Annahme ist natürlich im allgemeinen falsch, hat sich aber trotzdem als sehr nützlich erwiesen. Ein gebräuchliches n ist 3 (*Trigramme*); die obige Annahme liest sich dann für Trigramme als:

$$P(w_i \mid w_1, \ldots, w_{i-1}) = P(w_i \mid w_{i-2}, w_{i-1}) \tag{2}$$

Die Wahrscheinlichkeit einer Wortfolge mit Länge N für ein festes N (Gleichung 1) läßt sich dann unter der Trigramm-Annahme (Gleichung 2) wie folgt ausrechnen:

$$\begin{aligned} P(w_{1,N}) &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1,2}) \dots P(w_N \mid w_{1,N-1}) && (3) \\ &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1,2}) \dots P(w_N \mid w_{N-2,N-1}) && (4) \\ &= P(w_1)P(w_2 \mid w_1) \prod_{i=3}^{N} P(w_i \mid w_{i-2,i-1}) && (5) \end{aligned}$$

Um die Formel weiter zu vereinfachen, führt man zwei Pseudoworte w_{-1} und w_0 ein (quasi Textanfangsindikatoren), so daß sich die Gleichung kürzer schreiben läßt als:

$$P(w_{1,N}) = \prod_{i=1}^{N} P(w_i \mid w_{i-2,i-1}) \tag{6}$$

Die Trigrammwahrscheinlichkeiten $P(w_i \mid w_{i-2,i-1})$ schätzt man ab, indem man in einem (genügend großen) Trainingstext die Anzahl aller möglichen Wortpaare und -tripel zählt. Basierend auf einer Zählung Z berechnet sich dann die geschätzte Wahrscheinlichkeit $P_g(x)$ für ein Trigramm x als:

$$P_g(w_i \mid w_{i-2,i-1}) = \frac{Z(w_{i-2,i})}{Z(w_{i-2,i-1})} \tag{7}$$

Um also z. B. die Wahrscheinlichkeit zu schätzen, daß das Wort „Ereignis" nach „für das" auftaucht, zählen wir, wie oft das Tripel „für das Ereignis" in dem Trainingstest vorkommt und wie oft „für das", und teilen. Bei dieser Methode entsteht das Problem, daß Worttripel, die im Trainingstext nicht vorkommen, aber durchaus möglich sind, eine Wahrscheinlichkeit von 0 zugewiesen bekommen, was natürlich nicht gewünscht ist (*sparse data problem*). Um das zu umgehen, werden die Wahrscheinlichkeiten „geglättet", indem auch die geschätzten Bi- und Unigrammwahrscheinlichkeiten verwendet werden. Für die Terme aus Gleichung 6 wird dann der folgende Ausdruck verwendet:

$$P(w_i \mid w_{i-2,i-1}) = \lambda_1 P_g(w_i) + \lambda_2 P_g(w_i \mid w_{i-1}) + \lambda_3 P_g(w_i \mid w_{i-2,i-1}) \tag{8}$$

Hierbei sind λ_1, λ_2, λ_3 nichtnegative Konstanten, so daß $\lambda_1 + \lambda_2 + \lambda_3 = 1$ ist, die eine Gewichtung der geschätzten Wahrscheinlichkeiten der Uni-, Bi- und Trigramme darstellen. Welche Werte die λ_i annehmen sollen, um möglichst gute Ergebnisse zu liefern, kann man natürlich durch Experimentieren feststellen. Besser und einfacher ist es jedoch, Trigramme als einen Spezialfall von HMM aufzufassen, und die für diesen Formalismus vorhandenen Lernalgorithmen zur Parameterschätzung zu verwenden.

2.2 Hidden-Markov-Modelle

Ein *Markov-Modell* ist vergleichbar mit einem endlichen Automaten, bei dem die Zustandsübergänge nicht durch eine Eingabe, sondern durch Übergangswahrscheinlichkeiten bestimmt werden. Jede Transition besteht aus einem Startzustand, einem Zielzustand, einer Ausgabe und einer Wahrscheinlichkeit, daß dieser Übergang erfolgt. Da von einem Zustand durchaus mehrere Transitionen mit der gleichen Ausgabe zu unterschiedlichen Zielzuständen ausgehen können, ist klar, daß in solchen Fällen nicht alleine aufgrund einer beobachteten Ausgabefolge die durchlaufenen Zustände feststellbar sind. Können nur die Ausgaben, nicht aber die durchlaufenen Zustände beobachtet werden, spricht man von einem *Hidden-Markov-Modell* (HMM).

Das Trigramm-Modell läßt sich nun durch ein HMM darstellen, indem für jedes mögliche Wortpaar ein Zustand eingeführt wird. Von jedem Zustand geht für jedes mögliche Wort (also Element des Ausgabealphabets) eine Transition in den entsprechenden Folgezustand, und zwar mit einer gewissen Übergangswahrscheinlichkeit, die der bedingten Wahrscheinlichkeit $P(w_i \mid$

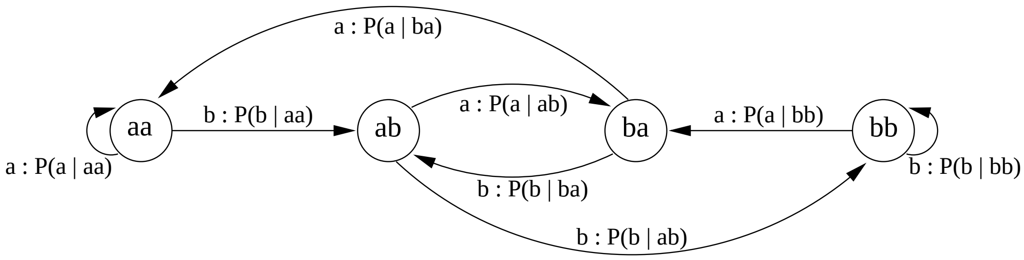

Abbildung 1: Ein HMM für Trigramme

$w_{i-2,i-1}$) aus dem Trigramm-Modell entspricht. Abbildung 1 zeigt ein HMM für das Ausgabealphabet $\{a, b\}$, das ein entsprechendes Trigramm-Modell simuliert.

Auch das verbesserte Trigramm-Modell, das Gleichung 8 zum Glätten der geschätzten Wahrscheinlichkeiten verwendet, läßt sich als HMM darstellen. Abbildung 2 veranschaulicht dies exemplarisch für einen kleinen Ausschnitt des HMMs aus Abbildung 1 (Zustand `ab` und dessen zwei Übergänge). Man mache sich klar, daß mit dem angegebenen HMM tatsächlich die bedingten Wahrscheinlichkeiten gemäß Gleichung 8 definiert werden: Wahrscheinlichkeiten entlang eines Pfades werden multipliziert, Wahrscheinlichkeiten verschiedener Pfade zwischen gleichen Anfangs- und Zielzuständen addiert.

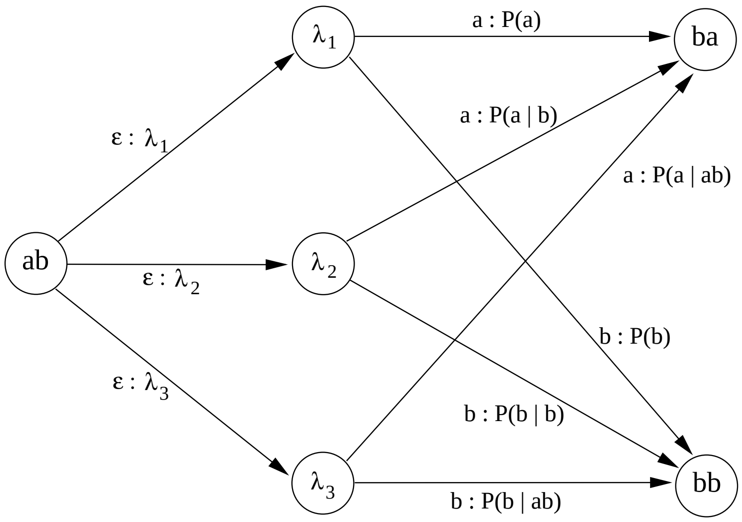

Abbildung 2: Ein HMM für das verbesserte Trigramm-Modell

Für HMM existieren Algorithmen, die den wahrscheinlichsten Pfad von Zuständen zu einer gegebenen Ausgabefolge (*Viterbi-Algorithmus*) oder die Wahrscheinlichkeit einer bestimmten Ausgabe berechnen (*Forward-Algorithmus*), und Trainingsalgorithmen, die die Übergangswahrscheinlichkeiten eines HMM so anpassen, daß die Wahrscheinlichkeit der Trainingssequenz maximiert wird (*Forward-Backward-* bzw. *Baum-Welch-Algorithmus*).

Mithilfe dieses Trainingsalgorithmus kann man ein HMM für das verbesserte Trigramm-Modell nach dem Vorbild von Abbildung 2 für eine gegebene Trainingssequenz automatisch optimieren.

2.3 Part-of-Speech Tagging

Eine weitere Anwendung von HMM ist die automatische Zuordnung der wahrscheinlichsten Folge von syntaktischen Kategorien (`N`, `V`, `Adj`, `Det`, etc.) zu einer Sequenz von Wörtern (*Part-Of-Speech Tagging* — POS-Tagging). Das Problem ist nicht so einfach, wie es klingt, hauptsächlich da sehr viele Wörter mehreren syntaktischen Kategorien angehören können. Betrachten wir beispielsweise den englischen Satz "I drove home", so kann "I" sowohl Nomen als auch Pronomen sein, "drove" Nomen und Verb und "home" Nomen, Verb und Adjektiv. Selbst bei diesem einfachen Satz gibt es also theoretisch zwölf verschiedene Möglichkeiten, den drei Wörtern POS-Tags zuzuordnen. Durch Verwendung von spezifischeren POS-Tags ergeben sich noch wesentlich mehr Ambiguitäten.

Formal läßt sich die Problemstellung des POS-Taggings ausdrücken als:

$$\arg\max_{t_{1,N}} P(t_{1,N} \mid w_{1,N}) = \arg\max_{t_{1,N}} \frac{P(w_{1,N}, t_{1,N})}{P(w_{1,N})} \tag{9}$$

$$= \arg\max_{t_{1,N}} P(w_{1,N}, t_{1,N}) \tag{10}$$

Mit $\arg\max_{t_{1,N}}$ wird dasjenige Argument $t_{1,N}$ (Folge von POS-Tags) bezeichnet, für das der dahinterstehende Ausdruck den maximalen Wert annimmt. Gleichung 9 benutzt die Definition der bedingten Wahrscheinlichkeit, in Gleichung 10 wird der Nenner weggelassen, da dieser für verschiedene $t_{1,N}$ konstant ist, bei der Bestimmung von $\arg\max_{t_{1,N}} P(t_{1,N} \mid w_{1,N})$ also irrelevant ist.

Bei der Modellierung eines HMM für das POS-Tagging werden die verschiedenen möglichen syntaktischen Kategorien den Zuständen zugeordnet, die Ausgabe des HMM besteht weiterhin aus den Wörtern der untersuchten Sprache und die Übergangswahrscheinlichkeit vom Zustand t_i zum Zustand t_{i+1} bei Ausgabe von w_i ist $P(w_i \mid t_i)P(t_{i+1} \mid t_i)$. D. h. man trifft (beispielsweise) die vereinfachenden Annahmen, daß die Ausgabe eines Wortes an einer bestimmten Stelle nur von dem an gleicher Stelle stehenden POS-Tag abhängt und daß das nächste POS-Tag (also der nächste Zustand) nur vom jeweils letzten beeinflußt wird. (Natürlich kann man auch andere Annahmen treffen, also mit einem anderen Sprachmodell arbeiten, aber das Grundprinzip des POS-Tagging mit HMM bleibt das gleiche.) Abbildung 3 zeigt ein Beispiel für ein HMM für POS-Tagging mit nur zwei möglichen syntaktischen Kategorien `N` und `V` und dem Ausgabealphabet $\{a, b\}$.

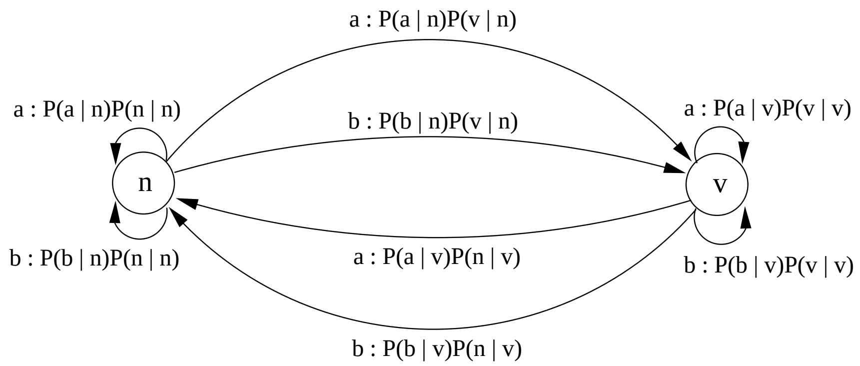

Abbildung 3: Ein HMM für Part-Of-Speech Tagging

Mit dem bereits im Abschnitt 2.2 erwähnten Algorithmus zur Bestimmung der wahrscheinlichsten Folge von durchlaufenen Zuständen (POS-Tags) bei gegebener Ausgabe (Wortfolge), läßt sich dann das POS-Tagging automatisch durchführen. Dabei ist die Rate der korrekt zugewiesenen syntaktischen Kategorien bei verbesserten Sprachmodellen, die dem hier vorgestellten ähnlich sind, im allgemeinen ungefähr 95 %.

3 Stochastische kontextfreie Grammatiken

Eine stochastische kontextfreie Grammatik (*Stochastic Context-Free Grammar* — SCFG) ist eine kontextfreie Grammatik erweitert um eine Wahrscheinlichkeitsverteilung P für die Produktionsregeln. Die Wahrscheinlichkeiten für alle Regeln, die das gleiche Nichtterminal expandieren, muß sich jeweils zu 1 aufsummieren. Die Wahrscheinlichkeiten für ein von der SCFG erzeugtes Wort (Kette von Terminalsymbolen), einen erzeugten Ableitungsbaum und eine Linksableitung werden wie folgt aufeinander aufbauend definiert:

- Die *Wahrscheinlichkeit einer Linksableitung* ist definiert als das Produkt der Wahrscheinlichkeiten der verwendeten Produktionsregeln; dem liegt die Annahme zugrunde, daß jede Anwendung einer Regel in einer Linksableitung von den zuvor verwendeten Regeln unabhängig ist.
- Die *Wahrscheinlichkeit eines Ableitungsbaums* ist gleich der Wahrscheinlichkeit der korrespondierenden Linksableitung.
- Die *Wahrscheinlichkeit eines Wortes* ist gleich der Summe aller Ableitungsbäume, die diese Terminalkette erzeugen.

Die Motivation, kontextfreie Grammatiken mit einer Wahrscheinlichkeitsverteilung zu erweitern, ist zum einen die dann mögliche Auflösung von syntaktischen Ambiguitäten. Vorausgesetzt, die Wahrscheinlichkeiten für die einzelnen Regeln wurden sinnvoll von Hand gesetzt oder automatisch gelernt (aus einem vorgeparsten Trainingskorpus). Dann kann aus mehreren möglichen Ableitungsbäumen eines zu parsenden Satzes der jeweils wahrscheinlichste selektiert werden. Zum anderen erlauben es SCFG, Grammatiken aus ausschließlich positiven Beispielen zu inferieren, indem wiederum ein vorgeparster Trainingskorpus benutzt wird, um die Wahrscheinlichkeiten der auftretenden Regeln zu schätzen. Das so entstehende Sprachmodell liefert allerdings schlechtere Ergebnisse als das Trigramm-Modell aus Abschnitt 2.1.

Das mag auf den ersten Blick erstaunlich sein, wird aber klarer, wenn man sich vergegenwärtigt, daß die Regeln einer SCFG in zwei Klassen zerfallen. Den Regeln, die die eigentliche grammatische Struktur wiedergeben (z. B. `S` $\rightarrow$ `VP NP`, `NP` $\rightarrow$ `Det Adj N`, usw.), stehen lexikalische Regeln gegenüber (`N` $\rightarrow$ banana, `N` $\rightarrow$ year, `Det` $\rightarrow$ the, `Adj` $\rightarrow$ green, usw.) Da die Anwendung einer Regel von den zuvor angewandten Regeln unabhängig ist, wird nach "the green" mit höherer Wahrscheinlichkeit "year" statt "banana" folgen, wenn "year" in dem Trainingskorpus häufiger vorkommt als "banana". Ein Trigramm-Modell dagegen würde der Wahrscheinlichkeit $P(\text{"banana"} \mid \text{"the green"})$ aller Voraussicht nach einen höheren Wert zuordnen als $P(\text{"year"} \mid \text{"the green"})$, da die Textsequenz "the green banana" in einem genügend großen Trainingskorpus sicherlich häufiger anzutreffen ist als "the green year".

Den SCFG fehlt also der lexikalische Kontext, den die Trigramm-Modelle berücksichtigen — deswegen heißen diese Grammatiken schließlich auch „kontextfrei“. Andererseits hat ein n-gramm-Modell keinerlei Möglichkeit, Abhängigkeiten von Wörtern zu berücksichtigen, die nicht zusammen in das Wortfenster der Breite n passen, oder hierarchische Strukturen einer Grammatik zu entdecken.

Dieses Dilemma aufzulösen versucht der im folgenden Abschnitt erläuterte Formalismus der stochastischen lexikalisierten Baumadjunktionsgrammatik von Yves Schabes ([Sch92]).

4 Stochastische lexikalisierte Baumadjunktionsgrammatiken

4.1 Motivation

Die am Ende des letzten Abschnitts erläuterten Probleme der Trigramm- bzw. Hidden-Markov-Modelle (keine hierarchischen Informationen, feste Breite des Wortfensters) und der SCFG (keine lexikalischen Informationen) brachten die Idee hervor, die Vorteile dieser beiden Ansätze zu vereinen, also ein hierarchisches Modell zur Verfügung zu haben, daß auch lexikalische Abhängigkeiten berücksichtigen kann.

In diesem und den beiden folgenden Abschnitten werden drei Formalismen vorgestellt, denen genau diese Zielsetzung zugrundeliegt. Alle drei benutzen lexikalisierte Baumadjunktionsgrammatiken (*Lexicalized Tree-Adjoining Grammar* — LTAG) als Ausgangspunkt. Nochmal zur Erinnerung: „Lexikalisiert“ heißt, daß jeder elementare (also initiale oder auxiliare) Baum der TAG mindestens ein Terminalsymbol als Blatt hat. Außer diesem sogn. *Anker* umfasst der Baum die minimalen syntaktischen und semantischen Einheiten, die von dem Anker verlangt werden. Ein elementarer Baum für ein Verb enthält beispielsweise Knoten, an denen die entsprechenden Argumente, die das Verb benötigt (Subjekt, Objekt, usw.), substituiert werden müssen. Abbildung 4 zeigt eine einfache LTAG mit initialen Bäumen $\alpha_1, \alpha_2, \alpha_3$ und auxiliaren Bäumen $\beta_1, \beta_2, \beta_3$. Die Substitutionsknoten sind mit $\downarrow$, die Fußknoten mit einem hochgestellten $*$ markiert. Die Indize an den Knotenbeschriftungen dienen der eindeutigen Identifizierung der einzelnen Knoten im weiteren Text, haben aber sonst keine Bedeutung.

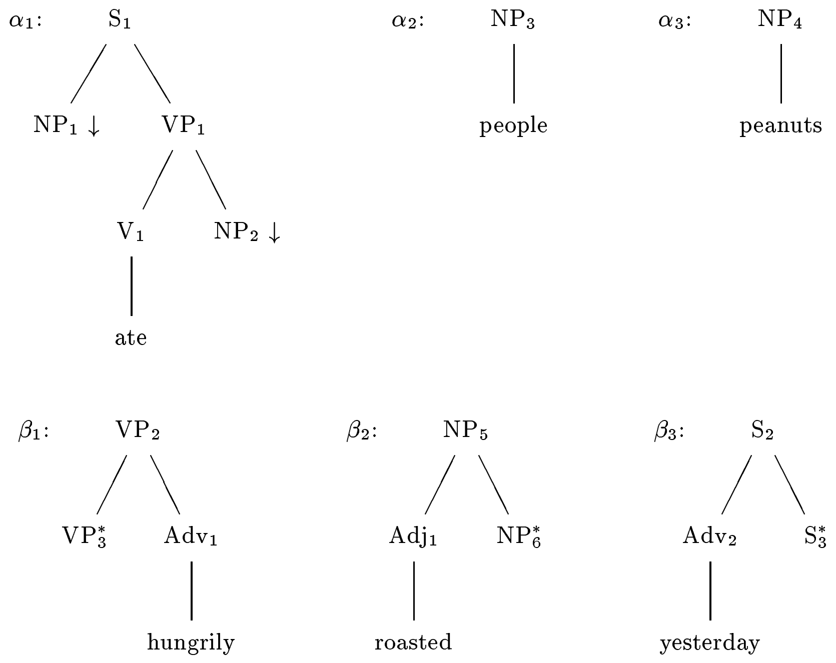

Abbildung 4: Elementare Bäume einer einfachen LTAG

LTAG erlauben es aufgrund ihrer erweiteren Lokalitätsdomäne und der Eigenschaft der Lexikalisierung, mithilfe einer Verteilungsanalyse der in einem Trainingskorpus vorkommenden Kombinationen dieser minimalen syntaktischen und semantischen Einheiten (den elementaren Bäumen) relevante statistische Aspekte der Sprache zu untersuchen, wie z. B. die Wahrscheinlichkeitsverteilung von Adverben, die in Zusammenhang mit einem bestimmten Verb auftauchen, welche Klassen von Wörtern als Argumente von prädikativen Elementen wie oft vorhanden sind, usw. Dabei ist vor allem das Beispiel der Ermittlung einer Wahrscheinlichkeitsverteilung von Adverben interessant, weil es nochmals verdeutlicht, daß dies mit n-gramm-Modellen oder SCFG nicht möglich ist. Adverben können beliebig weit von dem dazugehörigen Verb entfernt stehen ("John eats peanuts hungrily", "John eats roasted peanuts hungrily", "John eats roasted peanuts he bought in a shop of a friend of his former high school teacher who always tortured him with algebra hungrily.", usw.), deshalb ist ein Zusammenhang zwischen Verb und Adverb mit n-Gramm-Modellen selbst bei beliebig großem, aber festen n, nicht zu fassen. SCFG wiederum fehlt die lexikalische Information, also welches Wort als Adverb im Ableitungsbaum auftaucht.

Es folgt nun zunächst die formale Definition einer SLTAG, dann ein Algorithmus zur Berechnung der Wahrscheinlichkeit, daß ein bestimmter Satz von einer gegebenen SLTAG berechnet wird, ein Algorithmus, um die Parameter einer SLTAG in einem iterativen Verfahren zu verbessern, und anschließend ein Anwendungsbeispiel dieses Algorithmus, nämlich das automatische Inferieren einer Grammatik aus einem gegebenen Trainingskorpus.

4.2 Definition

Informal kann man SLTAG definieren, indem man dem Ereignis, daß ein bestimmter elementarer Baum mit einem bestimmten Knoten eines anderen elementaren Baums kombiniert wird (durch Adjunktion oder Substitution), eine Wahrscheinlichkeit zuordnet. Diese Kombinationen sind die stochastischen Ereignisse, die man betrachtet. Formal werden SLTAG definiert, indem sie auf stochastische linear indizierte Grammatiken (*Stochastic Linear Indexed Grammar* — SLIG) zurückgeführt werden, die wiederum um Wahrscheinlichkeitsverteilungen erweiterte linear indizierte Grammatiken (*Linear Indexed Grammar* – LIG) sind. Der Grund für diese etwas umständlich anmutende Art der Definition (eine intuitiv näherliegende Art der formalen Definition wird zum Vergleich kurz im Abschnitt 5 vorgestellt) ist die Ähnlichkeit von (S)LIG mit kontextfreien Grammatiken, die es ermöglicht, viele der bereits für CFG vorhandenen Algorithmen relativ einfach auf LIG zu übertragen. Diese können dann wiederum — wenn man die SLTAG in eine SLIG überführen kann — ohne große Änderungen auf SLTAG angewandt werden. Zum Beispiel ist der in Abschnitt 4.4 vorgestellte Algorithmus zur Schätzung der Parameter einer SLTAG eine Adaption eines ähnlichen Verfahrens für SCFG, das wiederum auf dem bereits erwähnten Forward-Backward-Algorithmus beruht.

4.2.1 Stochastische linear indizierte Grammatiken

Eine LIG ist in erster Näherung eine CFG, in der die Nichtterminale durch einen Keller (*Stack*) erweitert sind. Beim Anwenden einer Regel kann — zusätzlich zur Ersetzung des Nichtterminals auf der linken Seite der Regel durch den Ausdruck auf der rechten Seite — ein Kellersymbol gepusht oder gepoppt werden. Eine Regel wird durch das Nichtterminal auf der linken Seite und das oberste Symbol des zu diesem Nichtterminal gehörenden Kellers angestoßen.

In einer LIG kann der Keller, der mit dem Nichtterminal auf der linken Seite einer Regel assoziiert ist, nur auf höchstens einen Keller eines Nichtterminals der rechten Seite kopiert werden (im Gegensatz zu indizierten Grammatiken (*Indexed Grammar* — IG), die nicht derart eingeschränkt sind).

LIG sind schwach äquivalent zu TAG ([VS87]), d. h. zu einer gegebenen LIG G_{lig} exisitiert eine TAG G_{tag}, so daß $L(G_{lig}) = L(G_{tag})$ (die beiden Grammatiken erzeugen die gleiche Sprache), und umgekehrt.

Hier werden SLIG für den Spezialfall definiert, daß alle Regeln binär verzweigen.

Definition 1 *Eine* stochastische linear indizierte Grammatik *(Stochastic Linear Indexed Grammar — SLIG) G ist ein 6-Tupel* $(V_N, V_T, V_I, S, Prod, P)$ *mit:*

- V_T*: endliche Menge von Terminalsymbolen*
- V_N*: endliche Menge von Nichtterminalsymbolen*

- V_I: *endliche Menge von Kellersymbolen*
- $S \in V_N$: *Startsymbol*
- *Prod: endliche Menge von Regeln der Form*

$$\begin{array}{rcl} X_0[\$\rho_0] & \to & a \quad (11) \\ X_0[\cdot\cdot\rho_0] & \to & X_1[\cdot\cdot\rho_1] \quad X_2[\$\rho_2] \quad (12) \\ X_0[\cdot\cdot\rho_0] & \to & X_1[\$\rho_1] \quad X_2[\cdot\cdot\rho_2] \quad (13) \\ X_0[\$\rho_0] & \to & X_1[\$\rho_1] \quad X_2[\$\rho_2] \quad (14) \end{array}$$

Dabei sind $X_0, X_1, X_2 \in V_N$, $a \in V_T$, $\rho_0 \in V_I$ *und* $\rho_1, \rho_2 \in V_I^*$.

- *P: Wahrscheinlichkeitsverteilung, die jeder Regel* $X[\cdot\cdot\rho] \to \Delta \ \in Prod$ *eine Wahrscheinlichkeit* $0 \leq P(X[\cdot\cdot\rho] \to \Delta) \leq 1$ *zuordnet, so daß gilt:*

$$\forall X \in V_N, \forall \rho \in V_I : \sum_{\Delta} P(X[\cdot\cdot\rho] \to \Delta) = 1 \quad (15)$$

Ein paar Worte zur Notation: $[\cdot\cdot\rho]$ bezeichnet einen — möglicherweise unbegrenzten — Keller, dessen oberstes Element ρ ist und dessen restlicher Inhalt schematisch durch $\cdot\cdot$ gekennzeichnet wird, \$ bezeichnet das Kellerende und Δ eine beliebige rechte Seite einer Produktion.

$P(X[\cdot\cdot\rho] \to \Delta)$ bezeichnet die Wahrscheinlichkeit, daß die Regel $X[\cdot\cdot\rho] \to \Delta$ zur Anwendung ausgewählt wird aus der Menge der Regeln mit der gleichen linken Seite $X[\cdot\cdot\rho]$. Damit P eine Verteilung ist, muß die Summe der Wahrscheinlichkeiten aller Regeln mit dem gleichen Nichtterminal und dem gleichen obersten Kellersymbol auf der linken Seite 1 sein; formal wird dies durch Gleichung 15 ausgedrückt.

Eine Ableitung beginnt mit dem Startsymbol S, dem ein leerer Keller zugeordnet ist ($S[\$]$).

Definition 2 *Die* Sprache *einer SLIG ist definiert als:*

$$L = \left\{ w \in V_T^* \;\middle|\; S[\$] \stackrel{*}{\Rightarrow} w \right\} \quad (16)$$

Definition 3 *Die* Wahrscheinlichkeit einer Ableitung *ist definiert als das Produkt der Wahrscheinlichkeiten aller an der Ableitung beteiligten Regeln, wobei für die mehrfache Verwendung einer Regel in einer Ableitung je ein Term dem Produkt hinzugefügt wird.*

Definition 4 *Die* Wahrscheinlichkeit eines Satzes *ist definiert als die Summe der Wahrscheinlichkeiten aller Ableitungen des Satzes.*

4.2.2 Umwandlung LTAG in SLIG

Es wird nun die Konstruktion einer LIG G_{slig} aus einer gegebenen LTAG G_{tag} angegeben, so daß $L(G_{slig}) = L(G_{tag})$ gilt und jede Ableitung in G_{tag} einer eindeutigen Ableitung in G_{slig} entspricht (und umgekehrt). Zusätzlich wird jeder Ableitung in G_{slig} eine Wahrscheinlichkeit zugeordnet. Um die unten beschriebene Konstruktion einfach zu halten, wird angenommen, daß jeder Knoten eines elementaren Baumes in G_{tag} entweder ein Blattknoten (also ein Fuß- oder ein nicht leerer Terminalknoten) oder binär verzweigend ist. Die in 4.3 und 4.4 beschriebenen Algorithmen lassen sich jedoch für LTAG generalisieren, die nicht in dieser Normalform sind.

Konstruktion von G_{slig}

Die Nichtterminale von G_{slig} sind "top" (t) und "bottom" (b), die Terminalsymbole sind die gleichen wie die der G_{tag}, die Menge der Kellersymbole ist die Menge der Knoten (nicht Knotenbeschriftungen) aller elementarer Bäume der G_{tag} plus das Kellerendesymbol \$, das Startsymbol ist "top" (t).

Für alle Wurzelknoten η_0 der initialen Bäume, deren Wurzeln mit dem Startsymbol S beschriftet sind, werden die folgenden Startregeln zu der Menge der Produktionsregeln von G_{slig} hinzugefügt:

$$t[\$] \overset{p}{\rightarrow} t[\$\eta_0] \tag{17}$$

Diese Regeln besagen, daß jede Ableitung oberhalb eines Wurzelknotens mit Beschriftung S eines initialen Baums starten muß. Die Wahrscheinlichkeit, daß eine Ableitung mit einem initialen Baum beginnt, der mit einem bestimmten lexikalischen Element assoziert und dessen Wurzel η_0 ist, wird durch p angegeben.

Desweiteren werden für alle Knoten η eines elementaren Baums die folgenden Regeln generiert:

- Falls η_1 und η_2 die zwei Kinder eines Knotens η sind, so daß η_2 auf dem Rückgrat liegt (d. h. den Fußknoten subsumiert), füge folgende Regel hinzu:

$$b[\cdot\cdot\eta] \overset{p=1}{\rightarrow} t[\$\eta_1]\ t[\cdot\cdot\eta_2] \tag{18}$$

- Auf ähnliche Weise wird, falls η_1 und η_2 die zwei Kinder eines Knotens η sind, so daß η_1 auf dem Rückgrat liegt (d. h. den Fußknoten subsumiert), die folgende Regel hinzugefügt:

$$b[\cdot\cdot\eta] \overset{p=1}{\rightarrow} t[\cdot\cdot\eta_1]\ t[\$\eta_2] \tag{19}$$

- Falls η_1 und η_2 die zwei Kinder eines Knotens η sind, so daß keiner von ihnen auf dem Rückgrat liegt, füge folgende Regel hinzu:

$$b[\$\eta] \overset{p=1}{\rightarrow} t[\$\eta_1]\ t[\$\eta_2] \tag{20}$$

  Da mit 18, 19 und 20 unmittelbare Dominanzbeziehungen (*immediate domination links*) ausgedrückt werden, sind die dazugehörigen Wahrscheinlichkeiten jeweils 1.

- Falls η ein Knoten ist, der mit einem Nichtterminal beschriftet ist, und falls an η nicht obligatorisch adjungiert werden muß (*obligatory adjunction constraint*), kann es mit einer gewissen Wahrscheinlichkeit p vorkommen, daß an η keine Adjunktion vorgenommen wird. Deswegen füge die folgende Regel ein:

$$t[\cdot\cdot\eta] \overset{p}{\rightarrow} b[\cdot\cdot\eta] \tag{21}$$

- Falls η ein Knoten ist, an dem der auxiliare Baum β (mit Wurzelknoten η_r) adjungiert werden kann, füge die folgende Regel ein:

$$t[\cdot\cdot\eta] \overset{p}{\rightarrow} t[\cdot\cdot\eta\eta_r] \tag{22}$$

  Da die beiden an der Adjunktion beteiligten Bäume lexikalisiert sind, erlaubt diese Regel Aussagen über syntaktische Modifikationen, z. B. über Satzadverben ("*Apparently* John left"), Adverben, die VPs modifizieren ("John left *abruptly*"), Relativsätze in NPs ("The man *who left* was happy") oder sogar über Satzkomplemente ("*John thinks that* Harry is sick").

- Falls η_f der Fußknoten eines auxiliaren Baums β ist, der adjungiert wurde, muß die Ableitung an dem Knoten unter η_f fortgesetzt werden. Für diesen Fall wird die folgende Regel eingefügt:

$$b[\cdot\cdot\eta_f] \overset{p=1}{\rightarrow} b[\cdot\cdot] \tag{23}$$

  Die Wahrscheinlichkeit dieser Regel ist 1, da die Entscheidung, daß β adjungiert wird, schon in einer Regel der Form 22 getroffen wurde.

- Falls η_1 der Wurzelknoten eines initialen Baums ist, der an einen für Substitution markierten Knoten η substituiert werden kann, füge die folgende Regel ein:

$$t[\$\eta] \overset{p}{\rightarrow} t[\$\eta_1] \tag{24}$$

Hier ist p die Wahrscheinlichkeit, daß der initiale Baum mit Wurzel η_1 am Knoten η substituiert wird. U. a. korrespondiert die Wahrscheinlichkeit dieser Regel mit der Wahrscheinlichkeit, daß eine bestimmte Argumentposition mit einem lexikalisierten Baum gefüllt wird.

- Falls schließlich η als einziges Kind einen Knoten besitzt, der mit dem Terminalsymbol a beschriftet ist, muß die folgende Regel eingefügt werden:

$$b[\$\eta] \overset{p=1}{\rightarrow} a \tag{25}$$

Beispiel

Tabelle 1 zeigt die generierten Regeln für die LTAG in Abbildung 4. In der ersten Spalte ist jeweils die Konstruktionsregel angegeben, gemäß welcher die Produktion hinzugefügt wurde, die zweite Spalte enthält die Produktionsregel der SLIG und in der dritten Spalte sind die erzeugten Produktionen durchnumeriert. Die Wahrscheinlichkeiten sind frei gewählt, allerdings so, daß die konstruierte SLIG gemäß Definition 5 wohldefiniert ist.

Eine beispielhafte Ableitung der so konstruierten SLIG für den Satz "People ate roasted peanuts hungrily" ist in Tabelle 2 zu sehen. Die Wahrscheinlichkeit für diese Ableitung ist (auf sechs Stellen gerundet) 0,047152. Zum Vergleich enthält Abbildung 5 den abgeleiteten Baum der LTAG und den dazugehörigen Ableitungsbaum.

17	$t[\$]$	$\overset{1}{\rightarrow}$	$t[\$\mathrm{S}_1]$	S1
18	$b[\cdot\cdot\mathrm{NP}_5]$	$\overset{1}{\rightarrow}$	$t[\$\mathrm{Adj}_1]\ t[\cdot\cdot\mathrm{NP}_6]$	S2
	$b[\cdot\cdot\mathrm{S}_2]$	$\overset{1}{\rightarrow}$	$t[\$\mathrm{Adv}_2]\ t[\cdot\cdot\mathrm{S}_3]$	S3
19	$b[\cdot\cdot\mathrm{VP}_2]$	$\overset{1}{\rightarrow}$	$t[\cdot\cdot\mathrm{VP}_3]\ t[\$\mathrm{Adv}_1]$	S4
20	$b[\$\mathrm{S}_1]$	$\overset{1}{\rightarrow}$	$t[\$\mathrm{NP}_1]\ t[\$\mathrm{VP}_1]$	S5
	$b[\$\mathrm{VP}_1]$	$\overset{1}{\rightarrow}$	$t[\$\mathrm{V}_1]\ t[\$\mathrm{NP}_2]$	S6
21	$t[\cdot\cdot\mathrm{S}_1]$	$\overset{0.9}{\rightarrow}$	$b[\cdot\cdot\mathrm{S}_1]$	S7
	$t[\cdot\cdot\mathrm{VP}_1]$	$\overset{0.8}{\rightarrow}$	$b[\cdot\cdot\mathrm{VP}_1]$	S8
	$t[\cdot\cdot\mathrm{V}_1]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot\mathrm{V}_1]$	S9
	$t[\cdot\cdot\mathrm{NP}_3]$	$\overset{0.99}{\rightarrow}$	$b[\cdot\cdot\mathrm{NP}_3]$	S10
	$t[\cdot\cdot\mathrm{NP}_4]$	$\overset{0.7}{\rightarrow}$	$b[\cdot\cdot\mathrm{NP}_4]$	S11
	$t[\cdot\cdot\mathrm{VP}_2]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot\mathrm{VP}_2]$	S12
	$t[\cdot\cdot\mathrm{Adv}_1]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot\mathrm{Adv}_1]$	S13
	$t[\cdot\cdot\mathrm{NP}_5]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot\mathrm{NP}_5]$	S14
	$t[\cdot\cdot\mathrm{Adj}_1]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot\mathrm{Adj}_1]$	S15
	$t[\cdot\cdot\mathrm{S}_2]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot\mathrm{S}_2]$	S16
	$t[\cdot\cdot\mathrm{Adv}_2]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot\mathrm{Adv}_2]$	S17
	$t[\cdot\cdot\mathrm{VP}_3]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot\mathrm{VP}_3]$	S18
...	$t[\cdot\cdot\mathrm{NP}_6]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot\mathrm{NP}_6]$	S19
...	$t[\cdot\cdot\mathrm{S}_3]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot\mathrm{S}_3]$	S20
22	$t[\cdot\cdot\mathrm{S}_1]$	$\overset{0.1}{\rightarrow}$	$t[\cdot\cdot\mathrm{S}_1\mathrm{S}_2]$	S21
	$t[\cdot\cdot\mathrm{VP}_1]$	$\overset{0.2}{\rightarrow}$	$t[\cdot\cdot\mathrm{VP}_1\mathrm{VP}_2]$	S22
	$t[\cdot\cdot\mathrm{NP}_3]$	$\overset{0.01}{\rightarrow}$	$t[\cdot\cdot\mathrm{NP}_3\mathrm{NP}_5]$	S23
	$t[\cdot\cdot\mathrm{NP}_4]$	$\overset{0.3}{\rightarrow}$	$t[\cdot\cdot\mathrm{NP}_4\mathrm{NP}_5]$	S24
23	$b[\cdot\cdot\mathrm{VP}_3]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot]$	S25
	$b[\cdot\cdot\mathrm{NP}_6]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot]$	S26
	$b[\cdot\cdot\mathrm{S}_3]$	$\overset{1}{\rightarrow}$	$b[\cdot\cdot]$	S27
24	$t[\$\mathrm{NP}_1]$	$\overset{0.98}{\rightarrow}$	$t[\$\mathrm{NP}_3]$	S28
	$t[\$\mathrm{NP}_1]$	$\overset{0.02}{\rightarrow}$	$t[\$\mathrm{NP}_4]$	S29
	$t[\$\mathrm{NP}_2]$	$\overset{0.1}{\rightarrow}$	$t[\$\mathrm{NP}_3]$	S30
	$t[\$\mathrm{NP}_2]$	$\overset{0.9}{\rightarrow}$	$t[\$\mathrm{NP}_4]$	S31
25	$b[\$\mathrm{V}_1]$	$\overset{1}{\rightarrow}$	ate	S32
	$b[\$\mathrm{NP}_3]$	$\overset{1}{\rightarrow}$	people	S33
	$b[\$\mathrm{NP}_4]$	$\overset{1}{\rightarrow}$	peanuts	S34
	$b[\$\mathrm{Adv}_1]$	$\overset{1}{\rightarrow}$	hungrily	S35
	$b[\$\mathrm{Adj}_1]$	$\overset{1}{\rightarrow}$	roasted	S36
	$b[\$\mathrm{Adv}_1]$	$\overset{1}{\rightarrow}$	yesterday	S37

Tabelle 1: Die zu der LTAG in Abbildung 4 konstruierte SLIG

	$t[\$]$		
$\stackrel{1}{\Rightarrow}$	$t[\$\mathrm{S}_1]$	S1	
$\stackrel{0.9}{\Rightarrow}$	$b[\$\mathrm{S}_1]$	S7	Keine Adjunktion von β_3 an S_1
$\stackrel{1}{\Rightarrow}$	$t[\$\mathrm{NP}_1]\ t[\$\mathrm{VP}_1]$	S5	
$\stackrel{0.98}{\Rightarrow}$	$t[\$\mathrm{NP}_3]\ t[\$\mathrm{VP}_1]$	S28	Substitution von α_2 an NP_1
$\stackrel{0.99}{\Rightarrow}$	$b[\$\mathrm{NP}_3]\ t[\$\mathrm{VP}_1]$	S10	Keine Adjunktion von β_2 an NP_3
$\stackrel{1}{\Rightarrow}$	people $t[\$\mathrm{VP}_1]$	S33	
$\stackrel{0.2}{\Rightarrow}$	people $t[\$\mathrm{VP}_1\mathrm{VP}_2]$	S22	Adjunktion von β_1 an VP_1
$\stackrel{1}{\Rightarrow}$	people $b[\$\mathrm{VP}_1\mathrm{VP}_2]$	S12	
$\stackrel{1}{\Rightarrow}$	people $t[\$\mathrm{VP}_1\mathrm{VP}_3]\ t[\$\mathrm{Adv}_1]$	S4	Fußknoten erbt Keller
$\stackrel{1}{\Rightarrow}$	people $b[\$\mathrm{VP}_1\mathrm{VP}_3]\ t[\$\mathrm{Adv}_1]$	S18	
$\stackrel{1}{\Rightarrow}$	people $b[\$\mathrm{VP}_1]\ t[\$\mathrm{Adv}_1]$	S25	Adjungierter B. β_1 abgearbeitet
$\stackrel{1}{\Rightarrow}$	people $t[\$\mathrm{V}_1]\ t[\$\mathrm{NP}_2]\ t[\$\mathrm{Adv}_1]$	S6	
$\stackrel{1}{\Rightarrow}$	people $b[\$\mathrm{V}_1]\ t[\$\mathrm{NP}_2]\ t[\$\mathrm{Adv}_1]$	S9	
$\stackrel{1}{\Rightarrow}$	people ate $t[\$\mathrm{NP}_2]\ t[\$\mathrm{Adv}_1]$	S32	
$\stackrel{0.9}{\Rightarrow}$	people ate $t[\$\mathrm{NP}_4]\ t[\$\mathrm{Adv}_1]$	S31	Substitution von α_3 an NP_2
$\stackrel{0.3}{\Rightarrow}$	people ate $t[\$\mathrm{NP}_4\mathrm{NP}_5]\ t[\$\mathrm{Adv}_1]$	S24	Adjunktion von β_2 an NP_4
$\stackrel{1}{\Rightarrow}$	people ate $b[\$\mathrm{NP}_4\mathrm{NP}_5]\ t[\$\mathrm{Adv}_1]$	S14	
$\stackrel{1}{\Rightarrow}$	people ate $t[\$\mathrm{Adj}_1]\ t[\$\mathrm{NP}_4\mathrm{NP}_6]\ t[\$\mathrm{Adv}_1]$	S2	
$\stackrel{1}{\Rightarrow}$	people ate $b[\$\mathrm{Adj}_1]\ t[\$\mathrm{NP}_4\mathrm{NP}_6]\ t[\$\mathrm{Adv}_1]$	S15	
$\stackrel{1}{\Rightarrow}$	people ate roasted $t[\$\mathrm{NP}_4\mathrm{NP}_6]\ t[\$\mathrm{Adv}_1]$	S36	
$\stackrel{1}{\Rightarrow}$	people ate roasted $b[\$\mathrm{NP}_4\mathrm{NP}_6]\ t[\$\mathrm{Adv}_1]$	S19	
$\stackrel{1}{\Rightarrow}$	people ate roasted $b[\$\mathrm{NP}_4]\ t[\$\mathrm{Adv}_1]$	S26	Adjungierter B. β_2 abgearbeitet
$\stackrel{1}{\Rightarrow}$	people ate roasted peanuts $t[\$\mathrm{Adv}_1]$	S34	
$\stackrel{1}{\Rightarrow}$	people ate roasted peanuts $b[\$\mathrm{Adv}_1]$	S13	
$\stackrel{1}{\Rightarrow}$	people ate roasted peanuts hungrily	S35	

Tabelle 2: Beispiel für eine Ableitung der SLIG in Tabelle 1

Definition 5 *Die SLIG, wie sie oben konstruiert wurde, ist* wohldefiniert, *falls die folgenden Gleichungen für alle Knoten η gelten:*

$$P(t[\cdot\cdot\eta] \to b[\cdot\cdot\eta]) + \sum_{\eta_1} P(t[\cdot\cdot\eta] \to t[\cdot\cdot\eta\eta_1]) = 1 \tag{26}$$

$$\sum_{\eta_1} P(t[\$\eta] \to t[\$\eta_1]) = 1 \tag{27}$$

$$\sum_{\eta_0} P(t[\$] \to t[\$\eta_0]) = 1 \tag{28}$$

Gleichung 26 besagt, daß die Wahrscheinlichkeiten für jeden Knoten η, daß an ihm ein Knoten η_1 adjungiert wird, über alle möglichen Knoten η_1 summiert, plus die Wahrscheinlichkeit, daß an η nicht adjungiert wird, in der Summe 1 ergeben müssen. Der Ausdruck in 27 stellt eine äquivalente Wohlgeformtheitsbedingung für alle an η möglichen Substitutionen, Gleichung 28 für alle mit S beschrifteten initialen Bäumen, mit denen eine Ableitung starten kann.

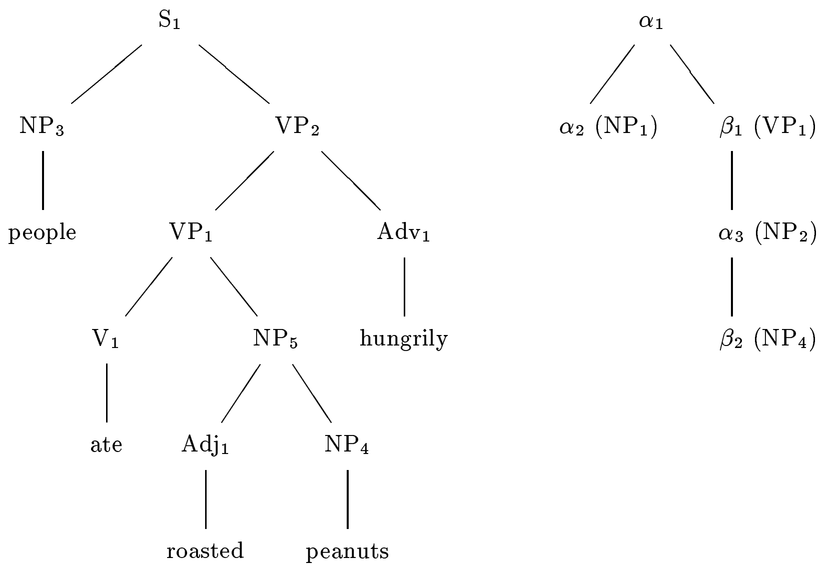

Abbildung 5: Abgeleiteter Baum (links) und Ableitungsbaum (rechts) der LTAG

Definition 6 *Eine Grammatik ist* konsistent, *falls sie folgende Gleichung erfüllt:*

$$\sum_{w \in \Sigma^*} P(t[\$] \stackrel{*}{\Rightarrow} w) = 1 \tag{29}$$

Durch die angegebene Konstruktion lassen sich auch Adjunktionsbeschränkungen (obligatorische, selektive oder Nulladjunktion) ausdrücken. Beispielsweise werden Adjunktionen blockiert, wenn man bei allen Regeln der Form 22 die Wahrscheinlichkeit auf 0 setzt.

4.3 Algorithmus zur Berechnung der Wahrscheinlichkeit eines Satzes

In diesem Abschnitt wird ein bottom-up-Algorithmus für SLTAG vorgestellt, der die Wahrscheinlichkeit eines Eingabesatzes berechnet. Der Algorithmus ist eine Erweiterung des CYK-Parsers für TAG ([VS87]). Der erweiterte Algorithmus parst immer größere Teilstücke der Eingabe und berechnet dabei deren Wahrscheinlichkeit.

Da die Zeichenkette entlang der Grenze eines auxiliaren Baums durch den Fußknoten in zwei Teilketten gesplittet wird, wird die Wahrscheinlichkeit betrachtet, daß ein Knoten zwei Teilketten der Eingabe generiert. Diese Entität wird innere Wahrscheinlichkeit (*inside probability*) genannt und weiter unten definiert.

Definition 7 *Eine* Teilfolge *der Eingabe* $w = a_1 a_2 \cdots a_N$ *von Position* i *bis* j *wird mit* w_i^j *bezeichnet und wie folgt definiert:*

$$w_i^j \stackrel{def}{=} \begin{cases} a_{i+1} \cdots a_j & \text{falls } i < j, \\ \epsilon & \text{falls } i \geq j. \end{cases} \tag{30}$$

Definition 8 *Sei eine Zeichenkette* $w = a_1 a_2 \cdots a_N$ *und eine SLTAG wie in (17 - 25) konstruiert gegeben. Dann ist die* innere Wahrscheinlichkeit $I^w(pos, \eta, i, j, k, l)$ *für alle Knoten* η *in einem elementaren Baum* α, *für* $pos \in \{t, b\}$ *und für alle Indize* $0 \leq i \leq j \leq k \leq l \leq N$ *wie folgt definiert:*

(i) Falls der Knoten η *nicht den Fußknoten von* α *subsumiert bzw. kein Fußknoten in* α *existiert, dann sind* j *und* k *nicht gebunden und:*

$$I^w(pos, \eta, i, -, -, l) \stackrel{def}{=} P(pos[\$\eta] \stackrel{*}{\Rightarrow} w_i^l) \tag{31}$$

(ii) Falls der Knoten η den Fußknoten η_f von α subsumiert, dann ist:

$$I^w(pos, \eta, i, j, k, l) \stackrel{def}{=} P(pos[\$\eta] \stackrel{*}{\Rightarrow} w_i^j \, b[\$\eta_f] \, w_k^l) \tag{32}$$

In (ii) spielt nur das oberste Element des Kellers eine Rolle, da aus $pos[\$\eta] \stackrel{*}{\Rightarrow} w_i^j \, b[\$\eta_f] \, w_k^l$ für alle Zeichenketten $\gamma \in V_I^*$ folgt, daß auch $pos[\$\gamma\eta] \stackrel{*}{\Rightarrow} w_i^j \, b[\$\gamma\eta_f] \, w_k^l$ gilt.

Algorithmus

Zur Initialisierung werden alle Wahrscheinlichkeiten auf 0 gesetzt. Dann wird die Berechnung von unten nach oben für immer größere Teilstücke der Eingabe fortgesetzt. Gestartet wird bei den Produktionen für lexikalische Einheiten:

1. Falls $b[\$\eta] \to a$ eine Produktion ist, dann ist

$$I^w(b, \eta, i, -, -, l) = \begin{cases} 1 & \text{falls } l = i+1 \wedge a = w_i^{i+1}, \\ 0 & \text{sonst.} \end{cases} \tag{33}$$

Dann werden die inneren Wahrscheinlichkeiten von immer größeren Teilketten der Eingabe bottom-up berechnet, wobei die folgenden Rekursionsgleichungen (*recurrence equations*) benutzt werden. Bezeichne dazu w die Eingabe, $\mathcal{F}$ die Menge der Fußknoten und $\mathcal{S}$ die Menge der Knoten, an denen substituiert werden kann. Dann kann man zeigen, daß für alle Knoten η eines elementaren Baums gilt:

2. Falls $\eta_f \in \mathcal{F}$, dann ist

$$I^w(b, \eta_f, i, j, k, l) = \begin{cases} 1 & \text{falls } i = j \wedge k = l, \\ 0 & \text{sonst.} \end{cases} \tag{34}$$

3. Falls $b[\cdot\cdot\eta] \to t[\cdot\cdot\eta_1] \; t[\$\eta_2]$, dann ist

$$I^w(b, \eta, i, j, k, l) = \sum_{m=k}^{l-1} I^w(t, \eta_1, i, j, k, m) \times I^w(t, \eta_2, m, -, -, l) \tag{35}$$

4. Falls $b[\cdot\cdot\eta] \to t[\$\eta_1] \; t[\cdot\cdot\eta_2]$, dann ist

$$I^w(b, \eta, i, j, k, l) = \sum_{m=i+1}^{j} I^w(t, \eta_1, i, -, -, m) \times I^w(t, \eta_2, m, j, k, l) \tag{36}$$

5. Falls $b[\$\eta] \to t[\$\eta_1] \; t[\$\eta_2]$, dann ist

$$I^w(b, \eta, i, -, -, l) = \sum_{m=i+1}^{l-1} I^w(t, \eta_1, i, -, -, m) \times I^w(t, \eta_2, m, -, -, l) \tag{37}$$

6. Für alle Knoten η, an denen adjungiert werden kann, setze

$$\begin{aligned} I^w(t, \eta, i, j, k, l) = & \\ & I^w(b, \eta, i, j, k, l) \times P(t[\cdot\cdot\eta] \to b[\cdot\cdot\eta]) \\ + & \sum_{r=i}^{j} \sum_{s=k}^{l} \sum_{\eta_1} I^w(t, \eta_1, i, r, s, l) \times I^w(b, \eta, r, j, k, s) \times P(t[\cdot\cdot\eta] \to t[\cdot\cdot\eta\eta_1]) \end{aligned} \tag{38}$$

7. Für alle Knoten $\eta \in \mathcal{S}$ setze

$$I^w(t,\eta,i,-,-,l) = \sum_{\eta_1} I^w(t,\eta_1,i,-,-,l) \times P(t[\$\eta] \to t[\$\eta_1]) \tag{39}$$

8. Für den Start der Ableitung setze

$$I^w(t,\$,i,-,-,l) = \sum_{\eta} I^w(t,\eta,i,-,-,l) \times P(t[\$] \to t[\$\eta]) \tag{40}$$

Wenn die inneren Wahrscheinlichkeiten vollständig berechnet sind, ergibt sich die *Wahrscheinlichkeit eines Satzes w* wie folgt:

$$P(w) \stackrel{def}{=} P(t[\$] \stackrel{*}{\Rightarrow} w) = I^w(t,\$,0,-,-,|w|) \tag{41}$$

Die Berechnung der inneren Wahrscheinlichkeiten für eine Eingabe der Länge N und eine Grammatik G benötigt im schlechtesten Fall $O(|G|^2 N^6)$ Zeit und $O(|G|N^4)$ Speicherplatz. (Dabei ist $|G|$ die Größe der Grammatik in einer geeigneten Darstellung.)

4.4 Algorithmus zum Verfeinern der Parameter einer SLTAG

Zu einer gegebenen Menge von positiven Beispielsätzen $W = \{w_1, w_2, \ldots, w_k\}$ möchte man die Wahrscheinlichkeiten der Regeln einer gegebenen SLTAG so berechnen, daß die Wahrscheinlichkeit, daß dieser Korpus W von der gegebenen SLTAG generiert wurde, maximiert wird. Den Algorithmus, der hier als Lösung vorgestellt wird, kann man auf zweierlei Arten verwenden.

Die eine Verwendungsmöglichkeit ist das Neuschätzen der Parameter einer SLTAG (*reestimation algorithm*). In diesem Fall generiert die Eingabe-SLTAG bereits Strukturen, die sinnvoll sind in bezug auf die verwendete linguistische Theorie und dem a priori vorhandenen Wissen über den gegebenen Korpus. Der Algorithmus verfeinert dann die Wahrscheinlichkeiten der einzelnen Regeln.

Die zweite mögliche Anwendung des Algorithmus ist als Lernalgorithmus. In einer ersten Iteration wird eine SLTAG benutzt, die alle möglichen Strukturen über einer gegebenen Menge von Knoten und Terminalsymbolen generiert. Anfangs sind die Wahrscheinlichkeiten aller Regeln zufällig gesetzt. Der Algorithmus wird dann mehrfach iteriert, um die Wahrscheinlichkeiten schrittweise zu verbessern.

Der Algorithmus startet mit einer ersten Schätzung der Parameter einer SLTAG; die von einem CYK-Parser geparsten Sätze des Trainingskorpus dienen als Basis zur Verfeinerung der Parameter. Nach jeder Iteration des Algorithmus, also jeder neuen Schätzung, hat sich die Wahrscheinlichkeit des Korpus erhöht, oder — äquivalent ausgedrückt — ist das Maß der Kreuzentropie (*cross entropy estimate*) kleiner geworden:

$$H(W,G) = -\frac{\sum_{w \in W} \log_2 P(w)}{\sum_{w \in W} |w|} \tag{42}$$

Um eine neue Schätzung herzuleiten, muß der Algorithmus für alle Sätze des Trainingskorpus die inneren Wahrscheinlichkeiten (siehe Definition 8) und die äußeren Wahrscheinlichkeiten berechnen.

Definition 9 *Sei eine Zeichenkette $w = a_1 a_2 \cdots a_N$ und eine SLTAG wie in (17 - 25) konstruiert gegeben. Dann ist die* äußere Wahrscheinlichkeit $O^w(pos,\eta,i,j,k,l)$ *für alle Knoten η in einem elementaren Baum α, für pos $\in \{t,b\}$ und für alle Indize $0 \le i \le j \le k \le l \le N$ wie folgt definiert:*

(i) Falls der Knoten η nicht den Fußknoten von α subsumiert bzw. kein Fußknoten in α existiert, dann sind j und k nicht gebunden und:

$$O^w(pos,\eta,i,-,-,l) \stackrel{def}{=} P(\exists\gamma \in V_I^*:\ t[\$] \stackrel{*}{\Rightarrow} w_0^i\, pos[\$\gamma\eta]\, w_l^N) \tag{43}$$

(ii) Falls der Knoten η den Fußknoten η_f von α subsumiert, dann ist:

$$O^w(pos,\eta,i,j,k,l) \stackrel{def}{=} P(\exists\gamma \in V_I^*:\ t[\$] \stackrel{*}{\Rightarrow} w_0^i\, pos[\$\gamma\eta]\, w_l^N \ \wedge\ b[\$\gamma\eta_f] \stackrel{*}{\Rightarrow} w_j^k) \tag{44}$$

Algorithmus

Sobald die inneren Wahrscheinlichkeiten berechnet sind, werden die äußeren Wahrscheinlichkeiten top-down berechnet, indem immer kleinere Teile der Eingabe betrachtet werden, angefangen mit $O^w(t,\$,0,-,-,N) = 1$ (per Definition). Sei also w wie gehabt der Eingabesatz, $\mathcal{W}$ die Menge der Wurzelknoten der initialen Bäume und $\mathcal{A}$ die Menge der mit Nichtterminalsymbolen beschrifteten Knoten auxiliarer Bäume. Dann gelten für alle Knoten η elementarer Bäume die folgenden Rekursionsgleichungen:

1. Falls $\eta \in \mathcal{W}$, dann ist
$$O^w(t,\eta,0,-,-,N) = P(t[\$] \to t[\$\eta]). \tag{45}$$
Außerdem gilt für alle $(i,j) \neq (0,N)$
$$O^w(t,\eta,i,-,-,j) = O^w(t,\eta_0,i,-,-,j) \times P(t[\$\eta_0] \to t[\$\eta]) \tag{46}$$

2. Falls η ein innerer Knoten ist, der den Fußknoten des elementaren Baums, zu dem er gehört, subsumiert, dann ist
$$\begin{aligned} O^w(t,\eta,i,j,k,l) = {} & \\ & \sum_{q=l+1}^{N} O^w(b,\eta_0,i,j,k,q) \times I^w(t,\eta_2,l,-,-,q) \times P(b[\cdot\cdot\eta_0] \to t[\cdot\cdot\eta]\, t[\$\eta_2]) \\ + & \sum_{p=0}^{i-1} O^w(b,\eta_0,p,j,k,l) \times I^w(t,\eta_1,p,-,-,i) \times P(b[\cdot\cdot\eta_0] \to t[\$\eta_1]\, t[\cdot\cdot\eta]) \end{aligned} \tag{47}$$

3. Falls η ein innerer Knoten ist, der den Fußknoten des elementaren Baums, zu dem er gehört, nicht subsumiert, dann ist
$$\begin{aligned} O^w(t,\eta,i,-,-,l) = {} & \\ & \sum_{q=l+1}^{N} O^w(b,\eta_0,i,-,-,q) \times I^w(t,\eta_2,l,-,-,q) \times P(b[\$\eta_0] \to t[\$\eta]\, t[\$\eta_2]) \\ + & \sum_{p=0}^{i-1} O^w(b,\eta_0,p,-,-,l) \times I^w(t,\eta_1,p,-,-,i) \times P(b[\$\eta_0] \to t[\$\eta_1]\, t[\$\eta]) \\ + & \sum_{j=l}^{N} \sum_{k=j+1}^{N} \sum_{q=k}^{N} O^w(b,\eta_0,i,j,k,q) \times I^w(t,\eta_2,l,j,k,q) \times P(b[\cdot\cdot\eta_0] \to t[\$\eta]\, t[\cdot\cdot\eta_2]) \\ + & \sum_{p=0}^{i-1} \sum_{j=p}^{i} \sum_{k=j}^{i} O^w(b,\eta_0,p,j,k,l) \times I^w(t,\eta_1,p,j,k,i) \times P(b[\cdot\cdot\eta_0] \to t[\cdot\cdot\eta_1]\, t[\$\eta]) \end{aligned} \tag{48}$$

4. Falls $\eta \in \mathcal{A}$, dann ist

$$
\begin{aligned}
&O^w(t,\eta,i,j,k,l) = \\
&\qquad \sum_{\eta_0} \sum_{p=j}^{k-1} \sum_{q=p+1}^{k} O^w(t,\eta_0,i,p,q,l) \times I^w(t,\eta_0,j,p,q,k) \times P(t[\cdot\cdot\eta_0] \rightarrow t[\cdot\cdot\eta_0\eta]) \\
&\quad + \sum_{\eta_0} O^w(t,\eta_0,i,-,-,l) \times I^w(t,\eta_0,j,-,-,k) \times P(t[\$\eta_0] \rightarrow t[\$\eta_0\eta])
\end{aligned} \tag{49}
$$

5. Falls η ein Knoten ist, der den Fußknoten des elementaren Baums, zu dem er gehört, subsumiert, dann ist

$$
\begin{aligned}
&O^w(b,\eta,i,j,k,l) = \\
&\qquad O^w(t,\eta,i,j,k,l) \times P(t[\cdot\cdot\eta] \rightarrow b[\cdot\cdot\eta]) \\
&\quad + \sum_{\eta_0} \sum_{p=0}^{i} \sum_{q=l}^{N} O^w(t,\eta,p,j,k,q) \times I^w(t,\eta_0,p,i,l,q) \times P(t[\cdot\cdot\eta_0] \rightarrow t[\cdot\cdot\eta_0\eta])
\end{aligned} \tag{50}
$$

6. Falls schließlich η ein Knoten ist, der den Fußknoten des elementaren Baums, zu dem er gehört, nicht subsumiert, dann ist

$$
\begin{aligned}
&O^w(b,\eta,i,-,-,l) = \\
&\qquad O^w(t,\eta,i,-,-,l) \times P(t[\$\eta] \rightarrow b[\$\eta]) \\
&\quad + \sum_{\eta_0} \sum_{p=0}^{i-1} \sum_{q=l}^{N} O^w(t,\eta,p,-,-,q) \times I^w(t,\eta_0,p,i,l,q) \times P(t[\$\eta_0] \rightarrow t[\$\eta_0\eta])
\end{aligned} \tag{51}
$$

Seien nun also die inneren und äußeren Wahrscheinlichkeiten gemäß den angegebenen Rekursionsgleichungen berechnet. Dann werden die Wahrscheinlichkeiten der Regeln der konstruierten SLIG angepaßt. Sei η ein Knoten in einem elementaren Baum, der den Fußknoten η_f im selben Baum subsumiert. Subsumiere außerdem η' den Fußknoten η'_f in einem (evtl. anderen) elementaren Baum. Die Formeln zum Verfeinern der Parameter für den Fall, daß η' an η adjungiert wird, und für den Fall, daß keine Adjunktion an η stattfindet, finden sich in den Gleichungen 52 und 53. Die anderen Fälle werden ähnlich behandelt.

$$
\hat{P}(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta']) = \frac{\sum_{w \in W} \frac{1}{P(w)} \times Q^w(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta'])}{\sum_{w \in W} \frac{1}{P(w)} \times \left[R^w(\eta) + \sum_{\eta'} Q^w(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta'])\right]} \tag{52}
$$

$$
\hat{P}(t[\cdot\cdot\eta] \rightarrow b[\cdot\cdot\eta]) = \frac{\sum_{w \in W} \frac{1}{P(w)} \times R^w(\eta)}{\sum_{w \in W} \frac{1}{P(w)} \times \left[R^w(\eta) + \sum_{\eta'} Q^w(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta'])\right]} \tag{53}
$$

Die in den Gleichungen benutzten Hilfsausdrücke Q^w und R^w werden wie folgt berechnet:

$$
\begin{aligned}
&Q^w(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta']) = \\
&\quad \sum_{i,r,j,k,s,l} P(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta']) \times I^w(t,\eta',i,r,s,l) \times I^w(b,\eta,r,j,k,s) \times O^w(t,\eta,i,j,k,l)
\end{aligned} \tag{54}
$$

$$
R^w(\eta) = \sum_{i,j,k,l} P(t[\cdot\cdot\eta] \rightarrow b[\cdot\cdot\eta]) \times I^w(t,\eta,i,j,k,l) \times O^w(b,\eta,i,j,k,l) \tag{55}
$$

(52) entspricht der durchschnittlichen Anzahl, daß die Regel $t[\cdot\cdot\eta] \to t[\cdot\cdot\eta\eta']$ benutzt wird, und (53) der durchschnittlichen Anzahl, daß kein Knoten an η adjungiert wird. Der Nenner von (52) und (53) schätzt die durchschnittliche Anzahl, wie oft eine Ableitung die Expansion von $t[\cdot\cdot\eta]$ enthält. Der Zähler von (52) schätzt die durchschnittliche Anzahl, wie oft eine Ableitung die Regel $t[\cdot\cdot\eta] \to t[\cdot\cdot\eta\eta']$ beinhaltet. Also ist (52) eine Schätzung der Wahrscheinlichkeit, daß die Regel $t[\cdot\cdot\eta] \to t[\cdot\cdot\eta\eta']$ benutzt wird.

Der Algorithmus iteriert, bis sich $H(W, G)$ nicht mehr verändert (innerhalb eines ϵ-Intervalls). Jede Iteration des Algorithmus benötigt im schlechtesten Fall $O(|G|N^6)$ Zeit für jeden Satz der Länge N.

4.5 Inferieren von Grammatiken mit SLTAG

Der in Abschnitt 4.4 dargestellte Algorithmus zum wiederholten Schätzen der Parameter einer SLIG wird nun dazu benutzt, eine Grammatik anhand eines gegebenen Trainingskorpus mit positiven Beispielen zu inferieren.

Die Anfangsgrammatik besteht aus allen Produktionsregeln einer SLIG für Bäume in lexikalisierter Normalform (*Lexicalized Normal Form* — LNF) über einer gegebenen Menge $\Sigma = \{a_i \mid 1 \leq i \leq T\}$ an Terminalsymbolen, wobei die Wahrscheinlichkeiten der Regeln auf 0 gesetzt sind (wir wissen ja am Anfang noch nichts über die zu lernende Grammatik). Bäume in LNF besitzen eine der drei in Abbildung 6 gezeigten Formen (die Indize an den Knoten dienen wiederum nur der eindeutigen Identifizierung und haben sonst keine Bedeutung). Grammatiken in dieser Normalform sind in der Lage, jede lexikalisierte Baumadjunktionssprache (*Tree-Adjoining Language* – TAL) zu generieren.

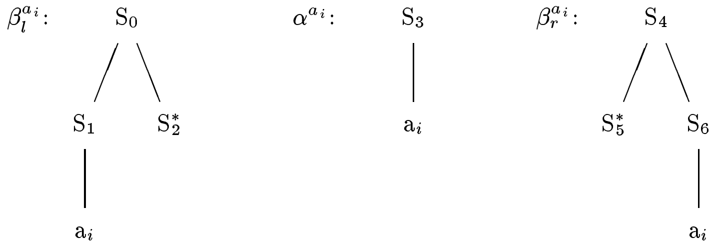

Abbildung 6: Baumformen in lexikalisierter Normalform

Die Zeitkomplexität von $O(N^6)$ des Algorithmus zum Verfeinern der Parameter macht diese Vorgehensweise zum Inferieren einer Grammatik in LNF im allgemeinen unpraktikabel. Beschränkt man sich allerdings auf Bäume der Formen $\beta_l^{a_i}$ und α^{a_i} (oder äquivalent der Formen $\beta_r^{a_i}$ und α^{a_i}), ist die generierte Sprache kontextfrei und der Algorithmus hat eine Zeitkomplexität von $O(N^3)$ im schlechtesten Fall.

Das System, das aus Bäumen der Formen $\beta_l^{a_i}$ und α^{a_i} besteht, kann man als stochastische lexikalisierte kontextfreie Grammatik (*Stochastic Lexicalized Context-Free Grammar* — SLCFG) betrachten, da es genau die kontextfreien Sprachen generiert, dabei aber noch lexikalische Informationen berücksichtigt.

Von Schabes durchgeführte Experimente ([Sch92]) mit der künstlichen Sprache $\{a^n b^n \mid n \geq 0\}$ und dem ATIS-Korpus (ATIS: "Air Traffic Information System") zeigen, daß in beiden Fällen die Kreuzentropie $H(W, G)$ beim Algorithmus für SLTAG schneller und zu einem besseren Wert konvergiert als beim entsprechenden Algorithmus für SCFG ([Bak79]).

4.6 Kritik

Die Einschränkung der Ausdrucksstärke der SLTAG von schwach kontextsensitiv auf kontextfrei macht nur für Sprachen wie beispielsweise das Englische Sinn, in denen diese zusätzliche

Mächtigkeit für eine adäquate linguistische Beschreibung nicht benötigt wird. In diesem Fall ist das dadurch gewonnene Mehr an Performanz Grundlage für die Anwendbarkeit des Algorithmus, Grammatiken von Grund auf aus einem Trainingskorpus zu lernen. Wie die von Schabes durchgeführten Experimente zeigen, ist durch die Lexikalisierung eine schnellere Konvergenz als bei SCFG möglich.

Für andere Sprachen, die die zusätzliche Mächtigkeit der schwachen Kontextsensitivität benötigen, scheint der Algorithmus nicht anwendbar, da zu langsam ($O(N^6)$ für jede Iteration und für jeden Satz der Länge N des Trainingskorpus). Trotzdem sind meines Erachtens auch in diesem Fall die SLTAG nützlich, um von Hand konstruierte Grammatiken mit Wahrscheinlichkeiten zu erweitern und so Informationen über die Relevanz von bestimmten linguistischen Phänomenen zu erhalten. Diese statistischen Daten können beispielsweise auch zur Desambiguierung von verschiedenen möglichen Ableitungsbäumen, die ein Parser für einen Eingabesatz konstruiert hat, nützlich sein.

Eine weitere kritische Betrachtung von SLTAG im Vergleich mit den anderen vorgestellten Methoden findet sich in Abschnitt 7.

5 Probabilistische Baumadjunktionsgrammatiken

Die im Abschnitt 4.2 vorgestellte Formalisierung des Gedankens, LTAG mit Wahrscheinlichkeiten zu erweitern, mag ein wenig umständlich erscheinen. Deshalb soll hier zum Vergleich kurz ein sehr ähnlicher Ansatz von Philip Resnik ([Res92]) vorgestellt werden, die probablistischen Baumadjunktionsgrammatiken (*Probabilistic Tree-Adjoining Grammar* — PTAG).

Die Definition von PTAG erfolgt durch eine direkte Zuordnung von Wahrscheinlichkeiten zu den in einer gegebenen LTAG möglichen Kombinationsoperationen (Adjunktion, Substitution), ohne den Umweg über einen anderen Formalismus zu gehen. Die Vorstellung dieser direkteren Formalisierung vertieft hoffentlich das Verständnis der **beiden** Formalismen zugrundeliegenden Konzepte.

Definition 10 *Zunächst ein paar Begriffsdefinitionen:*

- *Sei I die Menge der initialen und A die Menge der auxiliaren Bäume der gegebenen LTAG.*
- *Jeder Baum α der LTAG besitzt eine (möglicherweise leere) Untermenge seiner Blattknoten, an denen Substitution stattfinden kann. Bezeichne diese Untermenge mit $s(\alpha)$.*
- *Adjunktion kann an jedem Knoten η in α erfolgen, der mit einem Nichtterminal beschriftet ist, solange $\eta \notin s(\alpha)$. Bezeichne diese Menge möglicher Adjunktionsknoten mit $a(\alpha)$.*
- *Bezeichne mit $S(\alpha, \alpha', \eta)$ das Ereignis, daß der Baum α' für den Knoten η im Baum α substituiert wird.*
- *Bezeichne mit $A(\alpha, \beta, \eta)$ das Ereignis, daß der auxiliare Baum β am Knoten η im Baum α adjungiert wird, und mit $A(\alpha, none, \eta)$ das Ereignis, daß keine Adjunktion an η stattfindet.*
- *Sei Ω die Menge aller Substitutions- und Adjunktionsereignisse.*

Definition 11 *Eine* probabilistische Baumadjunktionsgrammatik *ist ein Quintupel (I, A, P_I, P_S, P_A), wobei I und A wie oben definiert sind, P_I eine Funktion von I in das reelle Intervall* $[0,1]$ *ist und P_S und P_A Funktionen von Ω nach* $[0,1]$ *sind, so daß:*

1. $$\sum_{\alpha \in I} P_I(\alpha) = 1$$
2. $$\forall \alpha \in I \cup A: \ \forall \eta \in s(\alpha): \sum_{\alpha' \in I} P_S(S(\alpha, \alpha', \eta)) = 1$$
3. $$\forall \alpha \in I \cup A: \ \forall \eta \in a(\alpha): \sum_{\beta \in A \cup \{none\}} P_A(A(\alpha, \beta, \eta)) = 1$$

$P_I(\alpha)$ wird als die Wahrscheinlichkeit interpretiert, daß eine Ableitung mit dem initialen Baum α beginnt. $P_S(S(\alpha, \alpha', \eta))$ steht für die Wahrscheinlichkeit, daß α' für den Knoten η in α substituiert wird, $P_A(A(\alpha, \beta, \eta))$ für die Wahrscheinlichkeit, daß β an den Knoten η in α adjungiert wird, und $P_A(A(\alpha, none, \eta))$ für die Wahrscheinlichkeit, daß an η keine Adjunktion stattfindet. Damit entsprechen die Bedingungen 1 bis 3 den in Definition 5 aufgestellten Wohlgeformtheitsbedingungen für SLIG.

Eine Ableitung einer TAG wird beschrieben als ein initialer Baum, mit dem diese startet, und eine Folge von Substitutions- und Adjunktionsoperationen, die dann durchgeführt werden. Formal wird jede Operation als $op(\alpha_1, \alpha_2, \eta)$ mit $op \in \{S, A\}$ und der initiale Baum, mit dem die Ableitung startet, als α_0 geschrieben.

Definition 12 *Sei eine Ableitung* $\tau = <\alpha_0, op_1(\cdots), \ldots, op_n(\cdots)>$ *einer TAG gegeben. Dann ist die* Wahrscheinlichkeit der Ableitung τ *definiert als:*

$$P(\tau) = P_I(\alpha_0) \prod_{1 \leq i \leq n} P_{op_i}(op_i(\cdots)) \tag{56}$$

6 Supertagging

Supertagging ([Ban97]) ist ein Verfahren, daß den Wörtern eines Satzes strukturelle Informationen zuordnen will. Dazu wird das in Abschnitt 2.3 erläuterte POS-Tagging so erweitert, daß den Wörtern statt syntaktischen Kategorien elementare Strukturen einer lexikalisierten Grammatik zugeordnet werden. Diese elementaren Strukturen werden als *Supertags* bezeichnet, um sie von den Part-Of-Speech-Tags abzugrenzen.

Prinzipiell läßt sich das Verfahren auf jede lexikalisierte Grammatik übertragen; LTAG bieten sich an, weil sie Abhängigkeiten lokalisieren. In diesem Fall sind die oben erwähnten elementaren Strukturen die elementaren Bäume einer LTAG.

Ein Wort ist i. a. mit mehr als einem Supertag assoziert, sogar dann, wenn das Wort ein eindeutiges POS-Tag besitzt. Da aber nach dem Parsen jedem Wort nur genau ein Supertag zugeordnet sein sollte (vorausgesetzt, es gibt keine globale Ambiguität), muß ein LTAG-Parser einen großen Suchraum an Supertags durchsuchen, um jedem Wort das richtige Supertag zuzuordnen und den Parse beenden zu können. Die Komplexität erhöht sich außerdem, weil die Zahl der verwendeten Supertags normalerweise wesentlich höher ist als die der POS-Tags (ca. Faktor zehn in einem von Bangalore zitierten Beispiel).

Diese Supertag-Desambiguierung kann nun (dem Prinzip der POS-Tagger folgend) vollständig oder zu großen Teilen vor dem eigentlichen Parsing durchgeführt werden, indem lokale Abhängigkeiten benutzt werden, z. B. in Form von n-gramm-Modellen (siehe Abschnitt 2.1), die auf der Verteilung von Supertags in einem gegebenen LTAG-geparsten Korpus basieren. Darüber hinaus können, da die Supertags Informationen auch über weit entfernte Abhängigkeiten kodieren, Informationen über die Verteilung der Distanz eines Supertags zu seinen abhängigen Supertags benutzt werden.

Ist im Idealfalle eine vollständige Supertag-Desambiguierung erfolgt, ist der Parse fast fertig und der Parser muß „nur noch“ die einzelnen elementaren Bäume kombinieren; daher nennt man dieses Verfahren „Beinahe-Parsing“ (*almost parsing*).

Im folgenden werde ich einige verwendete Modelle und darauf basierende Experimente zur Supertag-Desambiguierung kurz vorstellen. Als Korpora wurden der “Air Traffic Information System”-Korpus (ATIS), Teile des “Wall Street Journal” (WSJ) und Bedienungsanleitungen von IBM (IBM) benutzt.

Verwendung von struktureller Information

Die Struktur eines Supertags stellt Beschränkungen auf, in welchen syntaktischen Umgebungen das Supertag benutzt werden kann. Ein einfaches Beispiel ist der (linke bzw. rechte) *Spann* eines Supertags, d. h. die minimale Anzahl von lexikalischen Einheiten, die ein Supertag (zur

Linken bzw. Rechten des Ankers) umfasst. Durch das Rausfiltern der Supertags, die aufgrund der Länge des vorliegenden Satzes und ihres Spanns nicht in Frage kommen, und durch das Entfernen von Supertags, bei denen das mit dem Supertag assozierte Wort nicht in dem Satz vorkommt, kann die durchschnittliche Anzahl von möglichen Supertags pro Wort schon von 47 auf 25 (also um knapp 50 %) reduziert werden (Testmenge: 2 012 WSJ-Sätze).

Unigramm-Modell

Zusätzlich zu der Verwendung von strukturellen Informationen wird im Unigramm-Modell die relative Häufigkeit von Supertags in Abhängigkeit von den mit ihnen assozierten Wörtern benutzt. Gesucht ist also nach Supertag($\mathrm{w_i}$) = $\arg\max_{t_i} P(t_i \mid w_i)$, wobei:

$$P(t_i \mid w_i) = \frac{fr(t_i, w_i)}{fr(w_i)} \tag{57}$$

Für Wörter, die im Trainingskorpus nicht auftauchen, wird statt des Wortes das POS-Tag benutzt. Die Genauigkeit lag bei zwei Experimenten auf verschiedenen Trainingsmengen bei 73,4 bzw. 77,2 %.

Trigramm-Modell

Die verwendeten Trigramm-Modelle wurden auf (POS-Tag, Supertag)-Paaren trainiert statt auf (Wort, Supertag)-Paaren, um den Einfluß von nur spärlich vorhandenen Daten (*sparse data problem*) zu verringern. Für (POS-Tag, Supertag)-Paare, die in der Trainingsmenge nicht vorkommen, wurde die Wahrscheinlichkeit geschätzt anhand der Verteilung der relativen Häufigkeiten der beobachteten Ereignisse im Korpus (*Good-Turing discounting technique*). Dieses „Glätten" (*smoothing*) der Wahrscheinlichkeitsverteilung verhindert, daß im Trainingskorpus nicht vorkommende Ereignisse die Wahrscheinlichkeit 0 erhalten.

Bei den durchgeführten Experimenten mit unterschiedlich großen Trainingsmengen aus dem WSJ lag der Prozentsatz der korrekt zugewiesenen Supertags zwischen 86 % (bei 8 000 Wörtern im Trainingskorpus) und 92,2 % (1 Million Wörter). Wurde das Trigramm-Modell auf Teilen des ATIS- bzw. IBM-Korpus trainiert, also auf eingeschränkteren Domänen, lag die Erkennungsrate auch bei vergleichsweise kleinen Trainingsmengen (14 000 Wörter im IBM-Korpus, 1 500 Wörter im ATIS-Korpus) bereits bei 90,3 % (IBM) bzw. 93,8 % (ATIS).

7 Fazit

Die Nachteile der Ansätze aus der *co-occurence statistics* (n-gramm-Modelle, HMM) sind, wie bereits erwähnt, daß nur lexikalische, aber keine hierarchischen Kontextinformationen in Betracht gezogen werden und daß nur der Kontext in einem Wortfenster fester Breite berücksichtigt wird. Dafür sind die verwendeten Algorithmen vergleichsweise schnell (quadratisch in der Satzlänge) und die zugrundeliegende Theorie ist gut verstanden. Typische Anwendungsgebiete, bei denen hierarchische Informationen nicht oder nur wenig gebraucht werden, sind POS-Tagging und Spracherkennung. Syntaktische Verarbeitung ist eingeschränkt möglich, beispielsweise das automatische Erkennen von einfachen Strukturen (*"chunking"*) wie z. B. Nominal- oder Präpositionalphrasen.

SLTAG erlaubt das Erheben von statistischen Informationen aus Korpora sowohl in bezug auf den lexikalischen Kontext als auch über hierarchische Abhängigkeiten, da die elementaren Bäume einer LTAG minimale syntaktische und semantische Einheiten sind. Der vorgestellte Algorithmus zum Schätzen der Wahrscheinlichkeit eines Satzes bei gegebener SLTAG ist mit $O(N^6)$ (N ist die Länge des Eingabesatzes) recht langsam. Noch gravierender ist dies beim Algorithmus zum wiederholten Schätzen der Parameter einer SLTAG, den man z. B. zum automatischen Inferieren einer Grammatik aus einer gegebenen Menge von positiven Beispielsätzen benutzen kann. Dabei ist die Laufzeit pro Iteration und pro Satz (der Länge N) $O(N^6)$, so daß die praktische Anwendung auf die Untermenge der SLCFG beschränkt bleibt, da dann die Laufzeit wieder kubisch ist.

Bei dieser Einschränkung auf SLCFG werden zwar aufgrund der Lexikalisierung bessere Ergebnisse als bei SCFG erzielt; für Sprachen, die sich nicht kontextfrei beschreiben lassen, eignet sich dieses Verfahren aber nicht.

PTAG stellen eine einfachere Formalisierung des Gedankens dar, TAG mit statistischen Informationen zu erweitern. Algorithmen wurden in dem Text von Resnik allerdings nicht erwähnt.

Supertagging schließlich erweitert das Verfahren des POS-Taggings mit seinen relativ schnellen Algorithmen und der gut verstandenen Theorie, indem statt POS-Tags die elementaren Bäume einer LTAG (die *Supertags*) den Wörtern zugeordnet werden. Dadurch ist die Verwendung von lexikalischen und strukturellen Informationen möglich. Nach erfolgter eindeutiger Zuordnung von Supertags zu den Wörtern des Eingabesatzes ist der Parse fast vollständig, da die einzelnen elementaren Bäume „nur noch" richtig kombiniert werden müssen. Die erzielte Genauigkeit bei der Supertag-Desambiguierung ist mit bis zu 92 bis 93 % recht hoch. Das Supertagging erlaubt allerdings keine statistischen Aussagen über die Kombination der den Wörtern zugeordneten elementaren Bäume.

Literatur

[Bak79] J. K. Baker. Trainable grammars for speech recognition. In Jared J. Wolf and Dennis H. Klatt, editors, *Speech communication papers presented at the 97th Meeting of the Acoustical Society of America*, MIT, Cambridge, MA, June 1979.

[Ban97] Srinivas Bangalore. *Complexity of Lexical Descriptions and its Relevance to Partial Parsing*. PhD thesis, University of Pennsylvania, 1997.

[Cha93] Eugene Charniak. *Statistical Language Learning*. The M. I. T. Press, Cambridge, Massachusetts/London, England, 1993.

[KS94] Brigitte Krenn and Christer Samuelsson. *The Linguist's Guide to Statistics*, 1994. To be obtained from `http://www.coli.uni-sb.de/~krenn/edu.html`.

[Res92] Philip Resnik. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *COLING-92*, volume 2, pages 418–424, 1992.

[Sch92] Yves Schabes. Stochastic lexicalized tree-adjoining grammars. In *COLING-92*, volume 2, pages 426–432, 1992.

[VS87] K. Vijay-Shanker. *A Study of Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania, 1987.

A Abkürzungsverzeichnis

ATIS	*Air Traffic Information System*
CFG	*Context-Free Grammar* Kontextfreie Grammatik
HMM	Hidden-Markov-Modell
IG	*Indexed Grammar* Indizierte Grammatik
LIG	*Linear Indexed Grammar* Linear indizierte Grammatik
LNF	*Lexicalized Normal Form* Lexikalisierte Normalform
LTAG	*Lexicalized Tree-Adjoining Grammar* Lexikalisierte Baumadjunktionsgrammatik
NLP	*Natural Language Processing* Verarbeitung natürlicher Sprache
POS	*Part-Of-Speech* Syntaktische Kategorie
PTAG	*Probabilistic Tree-Adjoining Grammar* Probabilistische Baumadjunktionsgrammatik
SCFG	*Stochastic Context-Free Grammar* Stochastische kontextfreie Grammatik
SLCFG	*Stochastic Lexicalized Context-Free Grammar* Stochastische lexikalisierte kontextfreie Grammatik
SLIG	*Stochastic Linear Indexed Grammar* Stochastische linear indizierte Grammatik
SLTAG	*Stochastic Lexicalized Tree-Adjoining Grammar* Stochastische lexikalisierte Baumadjunktionsgrammatik
TAG	*Tree-Adjoining Grammar* Baumadjunktionsgrammatik
TAL	*Tree-Adjoining Language* Baumadjunktionssprache
WSJ	*Wall Street Journal*
ZV	Zufallsvariable