

Vergleich verschiedener Taggingverfahren unter besonderer Berücksichtigung des Auftretens unbekannter Worte

Schriftliche Hausarbeit
für die Bachelorprüfung der Fakultät für Philologie
an der Ruhr-Universität Bochum
(Gemeinsame Prüfungsordnung für das Bachelor/Master-Studium
im Rahmen des 2-Fach-Modells an der RUB vom 7.1.2002)

vorgelegt von

Björn Wilmsmann
Stefan-George-Str. 15a
46117 Oberhausen
bjoern@wilmsmann.de

abgegeben am 15.02.2006

Erstgutachter: Prof. Dr. Tibor Kiss
Zweitgutachter: Dr. Martin Hoelter

Inhaltsverzeichnis

1. Part-of-Speech Tagging – Diverse Verfahren, ein Überblick (S. 2-4)
2. Methoden zum Part-of-Speech Tagging (S. 4-18)
 - 2.1 Tagging auf Basis von (Hidden) Markov Modellen (S. 4-9)
 - 2.1.1 Markov Modelle (S. 4-5)
 - 2.1.2 Hidden Markov Modelle (S. 5-6)
 - 2.1.3 Der Viterbi-Algorithmus (S. 7-9)
 - 2.2 T'n'T – Trigrams and Tags (S. 9-11)
 - 2.2.1 Markov Modell Zweiter Ordnung (S. 9-10)
 - 2.2.2 Glättungsverfahren (S. 10)
 - 2.2.2 Weitere Verbesserungen (S. 10-11)
 - 2.3 Constraint-Grammar-basierte Taggingverfahren (S. 11-14)
 - 2.4 Transformationsbasiertes Tagging (S. 14-16)
 - 2.5 Tagging mittels Maximum Entropy Model (S. 17-18)
3. Die vorgestellten Ansätze im Vergleich (S. 19-27)
 - 3.1 Verarbeitung unbekannter Worte (S. 19-24)
 - 3.1.1 Tagging auf Basis von (Hidden) Markov Modellen (S. 19-20)
 - 3.1.2 T'n'T – Trigrams and Tags (S. 20-22)
 - 3.1.3 Constraint-Grammar-basierte Taggingverfahren (S. 22)
 - 3.1.4 Transformationsbasiertes Tagging (S. 23)
 - 3.1.5 Tagging mittels Maximum Entropy Model (S. 24)
 - 3.2 Genauigkeit im Vergleich anhand des Corpus Gesproken Nederlands (S. 24-25)
 - 3.3 Laufzeiteffizienz (S. 26)
 - 3.4 Beobachtungs- und Beschreibungsadäquatheit (S. 26-27)
4. Zusammenfassung, Ausblick und Fazit (S. 28-29)
5. Literaturverzeichnis (S. 30)

Als *Part-of-Speech (POS) Tagging*, kurz *Tagging* genannt, bezeichnet man das Einordnen von Wörtern eines Korpus in bestimmte durch sogenannte *Tagsets*, wie z.B. dem STTS (Schiller et. al., 1999) für das Deutsche oder dem für das Penn Treebank Korpus des Englischen entworfenen Tagset (Jurafsky & Martin, 2004: 2), definierte Kategorien.

Das Ziel einer solchen Kategorisierung ist es, neben dem eigentlichen Korpus und den in ihm enthaltenen Worten auch Informationen zu den sprachlichen Kategorien bzw. den Wortarten dieser Worte zu kodieren, indem man jedes Wort mit einem *Tag* versieht.

Durch eine solche *Disambiguierung* von potentiellen *Lesarten* für ein bestimmtes Wort lassen sich zusätzliche Informationen gewinnen, die im Rahmen einer späteren syntaktischen oder semantischen Analyse, z.B. zur Spracherkennung, zur maschinellen Übersetzung oder zum Information Retrieval, genutzt werden können.

Neben der naheliegenden, aber bei den heutzutage, nicht zuletzt durch das Internet, vorhandenen umfangreichen Korpora nicht mehr praktikablen und nicht zuletzt mühseligen Methode, ein solches Tagging manuell durchzuführen, liegen mittlerweile zahlreiche sogenannte *Tagger* vor, welche diesen Vorgang mittels bestimmter Verfahren und Algorithmen weitestgehend automatisiert erledigen, respektive eine Unterstützung beim manuellen Tagging darstellen.

Im Folgenden soll es um die Darstellung und den Vergleich verschiedener dieser Verfahren gehen, wobei neben allgemeinen Betrachtungen wie Genauigkeit, Laufzeiteffizienz, Lernaufwand bei statistischen und manueller Arbeitsaufwand bei regelbasierten Verfahren ein besonderes Augenmerk auf die Behandlung von für den Tagger nicht bekannten, also nicht bereits in einem Trainingsdurchlauf gelernten bzw. nicht bereits im Lexikon vorhandenen Worten gelegt werden soll.

1. Part-Of-Speech Tagging – Diverse Verfahren, ein Überblick

Die mittlerweile recht zahlreichen vorhanden Tagger und die ihnen zugrunde liegenden Verfahren lassen sich in drei Kategorien einteilen:

- Tagger, die das ihnen zugrundeliegende Modell aufgrund von statistischen Methoden weitestgehend selbständig lernen
- regelbasierte Tagger mit einem manuell erzeugten Regelwerk
- Hybridformen, die auf Regelwerken basieren, diese jedoch weitestgehend selbständig erlernen

Zu der ersten Kategorie zählen zuvorderst die auf Markov- bzw. Hidden-Markov-Modellen (HMM) basierenden Verfahren. Hier sind z.B. solche Tagger zu nennen, die den Viterbi-Algorithmus benutzen. Bei diesem Algorithmus wird aus den *Übergangswahrscheinlichkeiten* eines Tags zu den vorhergehenden Tags und der Wahrscheinlichkeit, dass ein Tag ein vorliegendes Wort *emittiert*, d.h. dieses Wort die zu diesem Zeitpunkt wahrscheinlichste Möglichkeit für das Tag darstellt, die wahrscheinlichste Sequenz von Tags für eine Sequenz von Worten ermittelt. Ein Verfahren, das diesen Algorithmus - mit einigen Modifikationen – erfolgreich nutzt, ist das Trigrams'n'Tags oder kurz T'n'T Verfahren von Brants (2000). Ebenfalls zu dieser Kategorie gehört das Tagging mittels eines *Maximum Entropy Model*, wie es von Ratnaparkhi (1996) genauer beschrieben wurde.

Die zweite Kategorie umfasst Tagger, die sich eines manuell definierten Regelwerks, der *Constraint Grammar*, und einer dem eigentlichen Tagging vorangehenden Disambiguierung auf morphologisch-lexikalischer Ebene bedienen (Voutilainen, 1995).

In der dritten Kategorie befinden sich schließlich solche Methoden, die Hybridformen aus statistischen und regelbasierten Ansätzen darstellen. Diese Kategorie enthält z.B. sogenannte *transformationsbasierte* Verfahren, die zwar ebenfalls mit der Hilfe eines Regelwerks arbeiten, dieses Regelwerk jedoch im Rahmen vordefinierter Vorgaben anhand der Auswirkungen, welche die Anwendung einer bestimmten Regel auf die Genauigkeit des Tagging-Resultats hat, überwacht lernen (Brill, 1995). Diese Verfahren stellen also deswegen eine Mischform dar, weil sie zwar zu den regelbasierten Verfahren gehören, aber sich diese Regeln eben aufgrund von mathematischen Verfahren selbst erschließen.

Allen Verfahren ist gemein, dass sie sich an ihrer Performanz, welche sich in den als *Precision* und *Recall* bezeichneten Werte widerspiegelt, messen lassen müssen (Voutilainen, 1995: 172):

- Recall: The ratio 'received appropriate readings / intended appropriate readings'
- Precision: The ratio 'received appropriate readings / all received readings'

Der Begriff Recall bedeutet also den Quotienten der korrekt erkannten Lesart durch die tatsächlich korrekten Lesarten, eine Rate von weniger als 100% bedeutet also hier, dass einige Worte nicht korrekt in ihrer Lesart erkannt wurden. Die Recall Rate eines Taggers wird auch als dessen *Genauigkeit* bezeichnet.

Als Precision wird nun die Rate bezeichnet, die sich aus dem Quotienten von während des Vorgangs korrekt erkannten Lesarten durch alle während des Vorgangs erkannten Lesarten ergibt, ein Wert von weniger als 100% für diesen Wert bedeutet, dass noch einige überflüssige Lesarten nicht verworfen wurden, die somit noch als 'Rauschen' in den Daten verbleiben (Voutilainen, 1995: 172).

Diese Werte sind neben dem verwendeten Verfahren stark abhängig davon, wieviele für den Tagger unbekannte Worte in einem Korpus (vor allen Dingen in Korpora, die viele fachsprachliche Elemente enthalten, ist das Vorkommen von Worten, die nicht bereits anhand eines Trainingskorpus gelernt werden konnten, naturgemäß recht hoch) vorkommen und welche Verfahren benutzt werden, um diese Worte dennoch zu verarbeiten und ihre Lesarten weitestgehend zu disambiguieren.

In Abschnitt 2 soll es zunächst darum gehen, einige ausgewählte und hier bereits kurz vorgestellte Verfahren genauer zu umreißen und diese daraufhin in Abschnitt 3 anhand verschiedener Kriterien, unter anderem eben der Behandlung unbekannter Worte, zu vergleichen.

2. Methoden zum Part-of-Speech Tagging

2.1 Tagging auf Basis von (Hidden) Markov Modellen

2.1.1 Markov Modelle

In der maschinellen Sprachverarbeitung, und dort insbesondere beim Part-of-Speech Tagging nehmen Verfahren, welche auf *Hidden Markov Modellen* basieren, einen prominenten Platz ein.

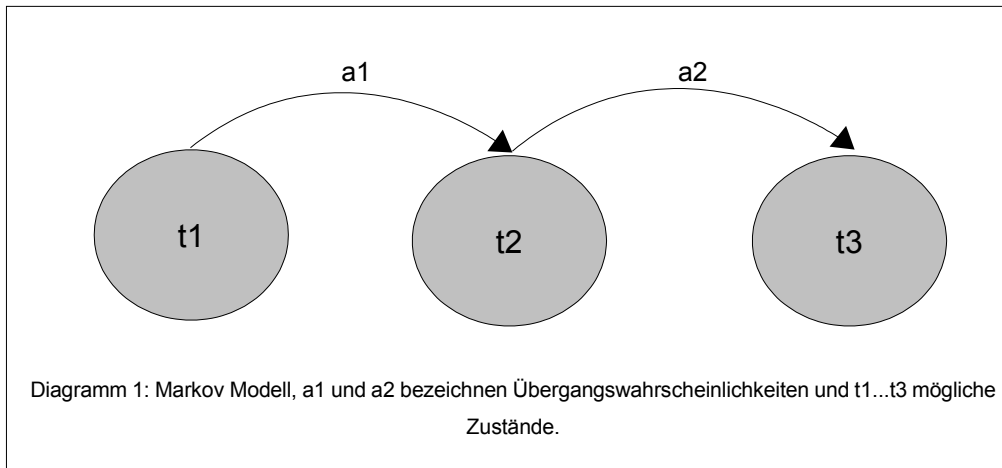
Hidden Markov Modelle sind, wie der Name bereits andeutet, Spezialfälle der nach dem Mathematiker Andrei Markov benannten *Markov Modelle*. Diese modellieren Abfolgen von Zuständen (im Falle des Part-of-Speech Tagging sind dies Tags) und deren Wahrscheinlichkeiten über einen Zeitraum T , wobei sie den folgenden Eigenschaften genügen (Manning & Schütze, 1999: 318):

- **Limited Horizon:** $P(X_{t+1} = s_k | X_1, \dots, X_t) = P(X_{t+1} = s_k | X_t)$
- **Time invariant:** $P(X_{t+1} = s_k | X_t) = P(X_2 = s_k | X_1)$

Die 'Limited Horizon' Eigenschaft beinhaltet, dass die Wahrscheinlichkeit für einen Zustand X_{t+1} ausschließlich vom direkten Vorzustand X_t abhängt, der vorherige Pfad also für diesen Wert nicht von Bedeutung ist. In bestimmten Erweiterungen von Markov Modellen, wird dieser begrenzte Horizont auf zwei oder mehr Vorzustände ausgedehnt (siehe z.B. das in 2.2 beschriebenen $T'n'T$ Verfahren).

Die Eigenschaft 'Time invariant' besagt des Weiteren, dass die Wahrscheinlichkeit für einen bestimmten Zustand vom Zeitpunkt, an dem dieser Zustand auftritt, unabhängig ist, sprich: Wenn die Wahrscheinlichkeit für einen Zustand X unter Voraussetzung eines Vorzustandes Y für den Zeitpunkt T_1 einen bestimmten Wert annimmt, so nähme sie den gleichen Wert für den Zeitpunkt T_2 an, falls dessen Vorzustand ebenfalls der gleiche wie zum Zeitpunkt T_1 ist.

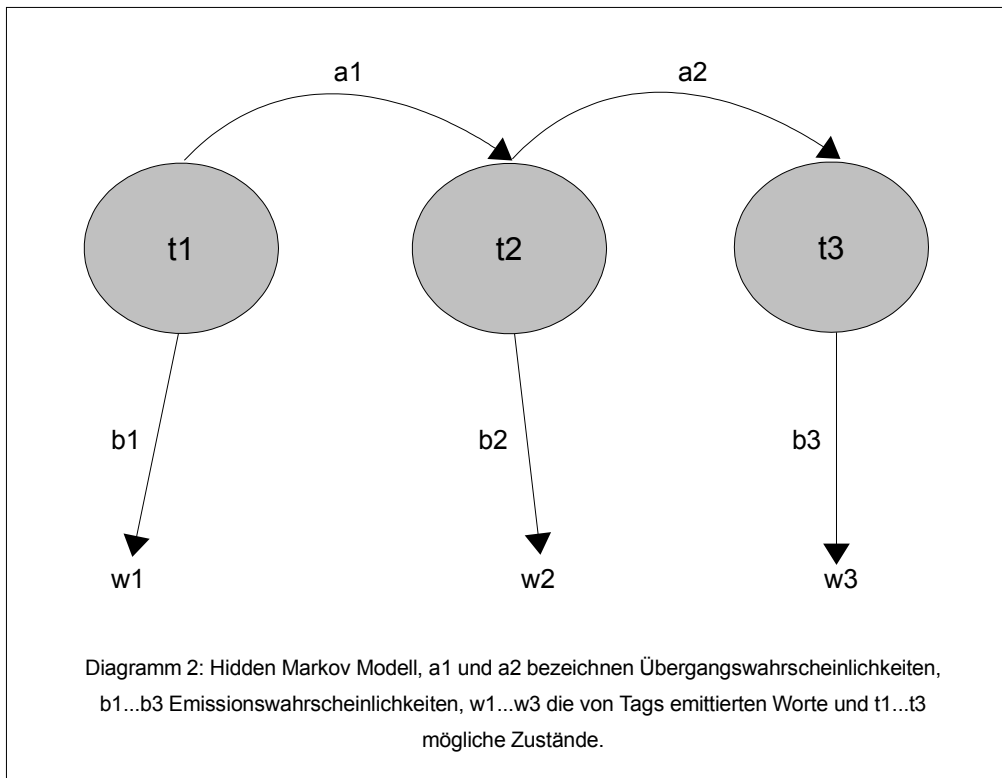
Aus diesen Bedingungen ergibt sich, dass es sich bei Markov Modellen um endliche Automaten handelt, weil eben jeder Zustand ausschließlich vom Vorzustand abhängt und keine Abhängigkeiten über größere Distanzen hinweg bestehen. Die Wahrscheinlichkeiten aller von einem Zustand zu möglichen Folgezuständen ausgehenden Pfade ergeben dabei immer 1.0 (Manning & Schütze, 1999: 317-319):



2.1.2 Hidden Markov Modelle

Für die Anforderungen, die beim Part-of-Speech Tagging an ein zugrundeliegendes Modell gestellt werden, bedarf es nun der Erweiterung auf Hidden Markov Modelle (Jurafsky & Martin, 2004: 27-30), weil mit einfachen Markov Modellen zwar die Übergangswahrscheinlichkeiten von Tags untereinander dargestellt werden können (Manning & Schütze, 1999: 351), die Wahrscheinlichkeiten von Tags für im Korpus auftretende Worte aber nicht repräsentiert sind.

Um dies nun ebenfalls zu berücksichtigen, wird in der Erweiterung des Modells davon ausgegangen, dass jedes Tag Worte mit einer bestimmten Wahrscheinlichkeit *emittiert*, also nicht etwa Worte Tags erzeugen, sondern genau anders herum. Dieses Vorgehen basiert auf Bayes' Theorem, welches besagt, dass sich eine Wahrscheinlichkeit für einen Wert A in Abhängigkeit von B auch als Wahrscheinlichkeit für den Wert B in Abhängigkeit von A ausdrücken lässt. Die Wahrscheinlichkeit, mit der ein bestimmtes Tag ein bestimmtes Wort emittiert, wird als *Emissionswahrscheinlichkeit* bezeichnet. Das folgende Diagramm verdeutlicht den Unterschied zu einfachen Markov Modellen:



Das konkrete Taggingverfahren auf Grundlage von Markov Modellen läuft nun in zwei Schritten ab. Zuerst müssen anhand eines bereits *tokenisierten*¹ und manuell *getaggtten* Korpus, des sogenannten *Trainingskorpus*, die Übergangswahrscheinlichkeiten von einem Zustand (bzw. Tag) zum nächsten, sowie die Wahrscheinlichkeiten, mit denen Worte von Tags emittiert werden, die sogenannten Emissionswahrscheinlichkeiten, gelernt werden. Hierzu wird zum einen die Anzahl der Ereignisse, bei denen ein Tag A vor einem Tag B auftritt, durch die Gesamtzahl der Ereignisse für Tag B und zum anderen die Ereignisse, bei denen einem Wort W ein Tag T zugeordnet wird, durch die Gesamtzahl der Ereignisse für Tag T dividiert. Dieser Vorgang wird daraufhin für alle Tags wiederholt (Manning & Schütze, 1999: 320-325, 345-349).

Nachdem sowohl Übergangs- als auch Emissionswahrscheinlichkeiten gelernt wurden, lässt sich das so erworbene Modell auf *ungetaggte* Korpora anwenden. Um nun aus dem Modell auf die korrekte Tagsequenz für eine gegebene Wortsequenz zu schließen, stehen verschiedene Algorithmen² zur Verfügung, das wohl bekannteste dieser Verfahren, der Viterbi-Algorithmus soll im Folgenden kurz vorgestellt werden.

1 Tokenisierung bezeichnet den Vorgang, bei dem ein Korpus in einzelne linguistische Entitäten, typischerweise Worte, unterteilt, Groß- und Kleinschreibung normalisiert und Satzzeichen von wortinternen Sonderzeichen disambiguiert werden, siehe dazu Halama (2004), Abschnitt 3.3.1.

2 Ein weiteres Verfahren wäre z.B. der Baum-Welch-Algorithmus, auch Forward-Backward-Algorithmus genannt (Manning & Schütze, 1999: 333)

2.1.3 Der Viterbi-Algorithmus

Der Viterbi-Algorithmus (Manning & Schütze, 1999: 349-351) lässt sich in drei Schritte unterteilen:

- 1.) Initialisierung
- 2.) Induktion bzw. vollständiges Durchlaufen der zu taggenden Sequenz
- 3.) Auslesen des Pfades

Im ersten Schritt wird die Wahrscheinlichkeit für ein Satztrennzeichen vor Beginn des Satzes auf 1.0 und für jedes andere Tag auf 0.0 gesetzt:

$$\delta_1(\text{PERIOD}) = 1.0$$

$$\delta_1(t) = 0.0 \text{ für jedes } t \neq \text{PERIOD}$$

Im Zuge des zweiten Schrittes werden zwei Funktionen für jedes $i \in [1 \dots n]$ iterativ berechnet:

- 1.) $\delta_{i+1}(j)$: Diese Funktion wählt für das Tag j das Maximum aus $[\delta_i(k) \times P(j|k)]$ für alle möglichen Tags k .
- 2.) $\psi_{i+1}(j)$: Diese Funktion liefert denjenigen Vorzustand k für das Tag j , von dem aus das mit $\delta_i(j)$ berechnete Maximum erreicht wurde.

Anhand des nun folgenden, in Python implementierten Code-Segmentes soll der Viterbi-Algorithmus näher erläutert werden:

```

def viterbi(corpus, tagset, startProbability, transitionProbabilities, emissionProbabilities):
    # go through each sentence in corpus
    for sentence in corpus:
        # initialization
        tagSequence = {}
        argMaxSequence = {}
        prob = startProbability

        # go through each word in sentence
        for word in sentence:
            tags = {}
            argMax = 0
            probMax = 0

            # go through tags
            for tag in tagset:
                # go through tags for each tag
                for nextTag in tagset:
                    # delta
                    thisProb = prob
                    thisProb *= transitionProbabilities[tag][nextTag] * emissionProbabilities[tag][word]

                    # if this delta is larger than the current maximum
                    if thisProb > probMax:
                        # delta
                        probMax = thisProb

                        # psi
                        argMax = tag

                        # values for tag(i + 1)
                        tags[nextTag] = (argMax, probMax)

            # read-out
            probMax = 0
            for tagBuffer in tags:
                (arg, prob) = tags[tagBuffer]
                if prob > probMax:
                    probMax = prob
                    argMaxSequence[word] = arg

    # return tag sequence
    return argMaxSequence

```

Die Funktion `viterbi(corpus, tagset, startProbability, transitionProbabilities, emissionProbabilities)` bekommt ein bereits tokenisiertes, d.h. in Sätze und Worte unterteiltes Korpus, das Tagset, welches auf das Korpus angewendet werden soll, eine Startwahrscheinlichkeit, die trainierten Übergangswahrscheinlichkeiten zwischen den Tags und die Emissionswahrscheinlichkeiten der Tags für die einzelnen Worte übergeben.

Zuerst wird die Wahrscheinlichkeit für das Startelement mit 1.0 initialisiert (siehe Schritt 1) auf der vorherigen Seite).

Daraufhin wird über jeden Satz des Korpus und jedes Wort eines Satzes iteriert. Für jedes Wort werden alle Tags des Tagsets zunächst aufgerufen und in dieser Schleife noch einmal durchlaufen, um eine *Trellis*³ von jedem Tag zu jedem potentiell folgenden Tag aufzuspannen. Es werden also in der Schleife, welche das Tagset ein zweites Mal durchläuft, nämlich in der Schleife, die zuerst über das Tagset iteriert, alle möglichen Übergangspfade zwischen den Tags durchlaufen.

3 Eine Trellis ist eine surjektive $m:n$ -Abbildung, bei der jedes m zu einem Zeitpunkt $t = i$ auf jedes n zu einem Zeitpunkt $t = i + 1$ abgebildet wird, siehe dazu auch Abbildung 9.5 in Manning & Schütze (1999: 328).

Für jeden dieser Pfade werden dann die formal als $\delta_{i+1}(j)$ und $\psi_{i+1}(j)$ bezeichneten Funktionen ausgeführt. Hierzu wird das Produkt aus der Wahrscheinlichkeit für den bisher wahrscheinlichsten Pfad, der Übergangswahrscheinlichkeit vom aktuellen Tag zu einem möglichen folgenden Tag und der Emissionswahrscheinlichkeit des aktuellen Tags für das gerade aktuelle Wort gebildet. Falls dieser Wert über dem bisher höchsten ermittelten Wert probMax (= dem Resultat von $\delta_{i+1}(j)$) liegt, wird er zum neuen Wert für probMax und argMax (= dem Resultat von $\psi_{i+1}(j)$) wird auf das Tag aus der ersten Schleife gesetzt, welches das neue probMax verursacht hat. Dieser argMax -Wert, also dasjenige Tag, das den höchsten Wert für das folgende Tag erzeugt (d.h. das wahrscheinlichste Tag für das nachfolgende Tag und das aktuelle Wort darstellt) wird anschließend für jedes Wort ausgelesen und die so erzeugte Sequenz zurückgegeben⁴.

Die hier vorgestellten Markov Modelle und der Viterbi-Algorithmus fungieren als Grundlage für viele verfeinerte Tagging-Techniken wie z.B. dem T'n'T Verfahren von Thomas Brants (2000), um das es im folgenden Abschnitt gehen wird.

2.2 T'n'T – Trigrams and Tags

2.2.1 Markov Modell Zweiter Ordnung

Das Trigrams and Tags Verfahren, kurz T'n'T, basiert auf den im vorherigen Abschnitt vorgestellten Hidden Markov Modellen, jedoch wird die Limited Horizon Bedingung für Markov Modelle im Falle des T'n'T abweichend formuliert:

$$P(X_{t+1} = s_k | X_1, \dots, X_t) = P(X_{t+1} = s_k | X_t, X_{t-1})$$

Es werden also nicht bloß *Bigramme*, d.h. der gerade betrachtete und der ihm vorangehende Zustand, sondern zusätzlich dazu auch die sich aus dem aktuellen und den zwei vorhergehenden Zuständen ergebenden *Trigramme*, daher der Name des Verfahrens, betrachtet. Das dem Verfahren zugrundeliegende Markov Modell zweiter Ordnung unterscheidet sich dadurch von Markov Modellen einfacher Ordnung, dass es die wahrscheinlichste Tagsequenz unter Berücksichtigung von jeweils zwei Vorzuständen berechnet (Brants, 2000: 1):

$$\text{argmax}_{t_1 \dots t_T} \left[\prod_{i=1}^T P(t_i | t_{i-1}, t_{i-2}) P(w_i | t_i) \right] P(t_{T+1} | t_T)$$

4 Eine weitere, eher formale Darstellung des Viterbi-Algorithmus findet sich in Manning & Schütze (1999) auf Seite 350.

Für eine gegebene Sequenz von Worten $w_1 \dots w_T$ wird diejenige Tagsequenz ermittelt, für die sich der höchste Wert der Produkte aus den Übergangswahrscheinlichkeiten zwischen den innerhalb des Horizontes liegenden Zuständen und der Wahrscheinlichkeit ergibt, dass das momentan betrachtete Tag ein Wort emittiert. Zusätzlich wird dieses Produkt noch mit der Wahrscheinlichkeit für einen 'end-of-sequence marker' (Brants, 2000: 2) gegeben das letzte Tag der Sequenz multipliziert, da diese zusätzliche Einbeziehung der Wahrscheinlichkeit, dass ein Satzzeichen aus der Menge [!?!] dem letzten Tag folgt, die Taggingergebnisse verbessert (Brants, 2000: 2).

2.2.2 Glättungsverfahren

Zwei weitere Aspekte, welche in Brants (2000) als essentiell für POS Tagger auf Markov Modell Basis hervorgehoben werden, sind statistische *Glättungsverfahren* (engl. *Smoothing*) und die Behandlung unbekannter Worte. Der letztere Aspekt wird in Abschnitt 3.4.2 näher behandelt, hier soll es zunächst um die Beschreibung des verwendeten Glättungsverfahrens gehen.

Aufgrund des *Sparse Data Problems*⁵ lassen sich die Wahrscheinlichkeiten für Trigramme nicht direkt ableiten, da sonst etliche Wahrscheinlichkeiten unnötigerweise auf 0 gesetzt würden, sondern müssen mittels eines Glättungsverfahrens berechnet werden. Das T'n'T Verfahren setzt hierbei auf *lineare Interpolation* (Manning & Schütze, 1999: 218-221), die Wahrscheinlichkeit eines Trigramms wird dabei folgendermaßen ermittelt:

$$P(t_3|t_1, t_2) = \lambda_1 P(t_3) + \lambda_2 P(t_3|t_2) + \lambda_3 P(t_3|t_1, t_2)$$

Die Gewichtungsfaktoren $\lambda_1 \dots \lambda_3$ ergeben die Summe von 1.0, zudem wird bei T'n'T die *kontextunabhängige* Variante der linearen Interpolation verwandt, d.h. die Gewichtungsfaktoren sind für jedes Trigramm gleich, da das Sparse Data Problem wiederum eine individuelle Gewichtung für jedes einzelne Trigramm unmöglich macht.

Die Werte für diese Faktoren werden mittels *Deleted Interpolation* errechnet. Bei dieser Methode wird der Wert für $\lambda_1 \dots \lambda_3$ um die Wahrscheinlichkeit eines Trigrammes inkrementiert, je nachdem ob das Unigram oder das Bigram, welches das Unigram enthält, oder das Trigram, welches wiederum das Bigram enthält, wahrscheinlicher ist (Brants, 2000: 2).

5 Als Sparse Data Problem wird die Tatsache bezeichnet, dass vielfach, gerade bei komplexen Phänomenen, diese Phänomene nicht häufig genug in einem Korpus auftreten, um relevante statische Daten zu gewinnen.

2.2.3 Weitere Verbesserungen

T'n'T enthält neben dem besonderen Augenmerk auf Smoothing und der Behandlung unbekannter Worte noch weitere Methoden, welche zum einen der Verbesserung des Taggingergebnisses und zum anderen der Effizienzsteigerung des Laufzeitverhaltens dienen.

Um eine weitere Verbesserung des Taggingresultats zu erreichen, markiert T'n'T jedes Tag noch zusätzlich mit der Information, ob das entsprechende Wort groß oder klein geschrieben wird, was sich besonders in Sprachen wie dem Englischen auswirkt, in denen lediglich Eigennamen derart kodiert werden (Brants, 2000: 3).

Zur Verbesserung des Laufzeitverhaltens des Viterbi-Algorithmus wird der Beam Search Algorithmus benutzt. Dieses heuristische Verfahren berücksichtigt nur jene Werte für die Funktion δ , welche nicht kleiner sind als der größte bisher ermittelte Wert dividiert durch einen zuvor empirisch ermittelten Schwellenwert θ . Ein Wert von 1000 verdoppelt dabei ungefähr die Effizienz, ohne die Genauigkeit des Taggers zu beeinträchtigen, wobei die Geschwindigkeit von der Anzahl der unbekanntenen Worte und der Ambiguitäten beeinflusst wird (Brants, 2000: 4).

2.3 Constraint-Grammar-basierte Taggingverfahren

Im Gegensatz zu den bisher vorgestellten, statistischen Methoden beruhen solche, die eine sogenannte *Constraint Grammar* als zugrundeliegendes Modell verwenden, auf einem gänzlich anderen Ansatz. Diese im folgenden Abschnitt näher beschriebene Methode konstruiert kein statistisches Modell der Objektsprache, sondern folgt dem traditionellen Ansatz, ein von Menschen, also nicht maschinell, generiertes Regelwerk als Modell für die behandelte Sprache zu benutzen.

Dieses Regelwerk, das Aussagen darüber trifft, welche Wortarten in bestimmten Kontexten auftreten können bzw. dürfen, wird als Constraint Grammar bezeichnet, da die in ihm enthaltenen Regeln bzw. *Constraints* das Vorkommen bestimmter Wortarten in bestimmten *Kontexten* entweder erlauben oder aber für unmöglich erklären.

Diese Verfahrensweise, welche das Problem der Wortartenambiguität von einer morphologischen Perspektive aus angeht, wird von Karlsson (1995) näher umrissen und von Voutilainen (1995) anhand eines konkreten Modells, dem *ENGCG* (kurz für *English Constraint Grammar*) für das Englische, erläutert.

Karlsson (1995) führt für den Vorgang des Taggings mittels einer Constraint Grammar die folgenden Schritte an:

- 1.) Preprocessing
- 2.) Lexikonaktualisierung
- 3.) Morphologische Analyse
- 4.) Lokale morphologische Disambiguierung
- 5.) Syntaktische Disambiguierung

In Schritt 1.) werden dabei zunächst Aspekte wie Normalisierung von Groß- und Kleinschreibung, Erkennen von Satzzeichen, Erkennen der einzelnen Token, kurz also all das, was üblicherweise als Tokenisierung bezeichnet wird, berücksichtigt.

Bei Schritt 2.) werden dem Lexikon neue im gerade bearbeiteten Korpus vorkommende *Lexeme* hinzugefügt, um diese im Rahmen von Schritt 3.) ebenso wie die bereits im Lexikon vorhandenen Lexeme morphologisch analysieren zu können.

Diese Analyse geht mit Hilfe eines *Transducers* (Kempe, 1997), im konkreten Fall der ENGCG mit ENGTWOL (Morphological Transducer Lexicon Description of English), vorstatten, der schrittweise Zeichen einer Wortform aus dem Korpus als Eingabe konsumiert und die für die konsumierte Zeichenkette möglichen morphologischen Lesarten als Ausgabe zurückgibt. Diese möglichen morphologischen Lesarten pro Wortform werden als *Kohorte* (Karlsson, 1995: 46) dieser Wortform bezeichnet. Hier wird noch einmal der Unterschied zu den vorher erwähnten statistischen Verfahren deutlich: Wurden dort mögliche Kategorien für ein Wort als Wahrscheinlichkeiten für die sie repräsentierenden Tags abgebildet, so wird hier zunächst davon ausgegangen, dass alle Lesarten gleich wahrscheinlich sind.

Im Allgemeinen erhaltenen in diesem Schritt 35-45% der Worte mehrere mögliche Lesarten zugewiesen (Voutilainen, 1995: 170).

In Schritt 4.) folgt nun bereits eine erste, lokale Disambiguierung der erhaltenen Kohorten. Dabei werden aufgrund der Konfiguration der anderen Lesarten in der Kohorte unmögliche Lesarten aus der Kohorte entfernt. Eine solche lokale Disambiguierung ist vor allem in solchen Sprachen sinnvoll, welche, wie z.B. das Deutsche, über äußerst produktive Prozesse der Kompositabildung verfügen und bei denen daher der Transducer in Schritt 3.) in bestimmten Fällen *übergeneriert*.

Schließlich folgt in Schritt 5.) das eigentliche Parsing auf Basis der Constraint Grammar, also die syntaktische Disambiguierung. Bei diesem Vorgang wird zunächst eine Datei eingelesen, welche die vorher manuell formulierten Constraints enthält. Jedes dieser Constraints gibt Kontextbedingungen an, unter denen bestimmte Lesarten aus bestimmten Kohorten verworfen werden, weil diese Lesarten in dem gegebenen Kontext nicht grammatisch wären. Von Karlsson (1995: 58) werden nun 3 mögliche bzw. sinnvolle⁶ Varianten von Constraints angegeben:

6 Denkbar wären allerdings auch solche Constraintvarianten, die jeweils das Komplement der Funktionen von Variante a.), b.) und c.) zurückgeben.

- a.) **=!** : Ein Constraint dieser Variante wählt die angegebene Lesart als korrekt aus und verwirft alle anderen Lesarten, falls die Kontextbedingungen erfüllt sind. Falls die Kontextbedingungen nicht erfüllt sind, wird nur die angegebene Lesart verworfen.
- b.) **=!** : Ein Constraint dieser Variante wählt die angegebene Lesart als korrekt aus und verwirft alle anderen Lesarten, falls die Kontextbedingungen erfüllt sind. Falls die Kontextbedingungen nicht erfüllt sind, geschieht nichts.
- c.) **=0** : Ein Constraint dieser Variante verwirft die angegebene Lesart, wenn die Kontextbedingungen erfüllt sind.

Ein Constraint im Rahmen des ENGCG ist allgemein folgendermaßen aufgebaut:

TOKEN (=!!|=|=0) READING CONTEXT

TOKEN steht dabei für mögliche Worte aus dem Korpus, auf die dieser Constraint angewendet werden soll. Es kann einerseits allgemein gesetzt werden, z.B. auf '@w', was zur Folge hätte, dass dieser Constraint auf alle Worte angewandt wird. Andererseits kann es auch ein bestimmtes Wort enthalten, wodurch dieser Constraint dann auch nur bei Vorkommen des spezifizierten Wortes Anwendung findet.

Als nächstes folgt ein Modifikator, der eine der oben beschriebenen Constraintvarianten auswählt. Daraufhin folgt die Lesart READING aus der Kohorte des gerade behandelten Wortes, die der Constraint verarbeiten soll und schließlich endet der Constraint mit der Angabe der Kontextbedingungen CONTEXT, für welche die jeweilige durch den Modifikator ausgewählte Aktion des Constraints ausgelöst wird. Die Kontextbedingungen setzen sich aus Booleschen Operatoren (z.B. NOT), Positionsangabe relativ zur Position des gerade behandelten Wortes (z.B. -1 für einen Schritt links von der momentanen Position, 1 für einen Schritt rechts von der momentanen Position), Klammerung und Lesarten für die jeweiligen Positionen zusammen.

Es folgt ein konkretes Beispiel für ein Constraint aus dem ENGCG (Karlsson, 1995: 59):

(@w =0 VFIN (-1 TO))

Dieser Constraint besagt, dass bei jedem Wort die Lesart VFIN aus der Kohorte entfernt wird, falls das Wort direkt vor dem gerade behandelten Wort in seiner Kohorte eine Lesart mit der Stammform „to“ hat.

Momentan erreicht der ENGCG Tagger nach Anwendung der 1100 Constraints, auf die er zurückgreifen kann, je nach Textkategorie eine Precision von 93-97% und einen Recall von 99,7%. Nach Anwendung weiterer 200 heuristischer Constraints sinkt zwar letzterer Wert auf 99,6%, jedoch sind bereits 96-98% der Worte nach diesem zusätzlichen Schritt bereits

vollständig disambiguiert, die Precision ist also noch einmal erheblich gestiegen (Voutilainen, 1995: 186).

Zudem bleibt anzumerken, dass diese Werte beide theoretisch bis 100% ansteigen könnten, der Tagger also irgendwann ein perfektes Ergebnis liefern könnte, wenn nur genügend, bzw. die richtigen Constraints definiert werden.

2.4 Transformationsbasiertes Tagging

Sprachmodelle auf HMM-Basis repräsentieren linguistische Informationen nur indirekt, als statische Daten. Beziehungen und Phänomene wie z.B. Bindung, die über einen größeren Horizont als die zwei vorhergehenden Token operieren, werden von diesen nicht erfasst. Eine Erweiterung des Horizontes auf *Tetragramme*, *Pentagramme* usw. erscheint nicht praktikabel, da sich die dabei anfallende Parametermenge exponentiell erhöhen würde, wodurch sich das Sparse Data Problem, welches, wie wir beim T'n'T Verfahren gesehen haben, bereits bei Trigrammen sorgfältiger Berücksichtigung durch Glättungsverfahren bedarf, noch weiter verschlimmern würde (Manning & Schütze, 1999: 362).

An diesem Punkt setzt das von Brill (1995) vorgestellte *Transformation-Based Error-Driven Learning* an. Dieses Verfahren ist in der Lage, eine größere Menge sowohl an syntaktischen, wie auch an lexikalischen Relationen, darzustellen.

Bevor das Verfahren auf ungetaggte Korpora angewandt werden kann, müssen mittels eines Lernalgorithmus und anhand eines manuell getaggtten Korpus zunächst sogenannte Transformationen für die Objektsprache (bzw. Textkategorie) trainiert werden.

Dem Lernalgorithmus werden ein Lexikon, ein manuell getaggttes Trainingskorpus, sowie dasselbe Korpus, allerdings ungetaggt übergeben. Die Worte im ungetaggtten Korpus werden mit den jeweils am häufigsten für sie vorkommenden Tags (also den Unigrammwahrscheinlichkeiten $P(t|w_i)$) initialisiert. Diese Informationen über die jeweils häufigsten Tags für Worte sind im Lexikon verzeichnet und werden anhand bereits korrekt getaggtter Daten gewonnen.

Daraufhin konstruiert der Lernalgorithmus eine Liste von Transformationen, die das Taggingresultat des mit Unigrammwahrscheinlichkeiten initialisierten Korpus näher an das Ergebnis des manuellen Taggings desselben Korpus heranbringen. Eine Transformation, welche die Fehlerrate des initialen Taggings verglichen mit dem manuellen Tagging verbessert, wird in die Liste der später auf ungetaggte Korpora anzuwendenden Transformationen aufgenommen. Dabei werden die Transformationen nach ihrer Effizienz, d.h. der Fehlerkorrekturrate, geordnet. Eine Transformation, die z.B. 10% der Fehler korrigiert, wird weiter oben in die Liste eingeordnet als eine solche, die lediglich 5% der Fehler korrigiert und somit auch zuerst angewandt.

Transformationen setzen sich aus zwei Bestandteilen zusammen, einer die Transformation auslösenden Umgebung (*Triggering Environment*) und einer Regel, die besagt, in welcher Weise das Tag für ein Wort geändert wird, falls die Transformation angewandt wird (*Rewrite Rule*). Rewrite Rules haben das schlichte Format $t_1 \rightarrow t_2$, d.h. ein Tag wird in ein anderes überführt (Manning & Schütze, 1999: 361-365).

Mögliche Triggering Environments werden durch Schablonen (*Templates*) definiert, welche die Menge der zulässigen Transformationen einschränken. Diese Templates werden bei Brill (1995) zunächst in zwei Gruppen unterteilt, solche die nicht auf lexikalischen Beziehungen (d.h. solche die keinen Bezug zu bestimmten Worten nehmen) beruhen und solche die lexikalische Beziehungen berücksichtigen. Später werden noch Templates hinzugefügt, welche der Behandlungen unbekannter Worte dienen, diese sollen in Abschnitt 3.1.4 näher beschrieben werden. Zuerst folgen nun die Templates aus der zuerst genannten Gruppe (Brill, 1995: 16-17):

„Verändere Tag a nach Tag b bei folgendem Triggering Environment:“

- 1.) Das vorhergehende/nachfolgende Wort ist mit dem Tag x versehen.
- 2.) Das Wort zwei Schritte vor/nach dem aktuellen Wort ist mit dem Tag x versehen.
- 3.) Eines der zwei vorhergehenden/nachfolgenden Worte ist mit Tag x versehen.
- 4.) Eines der drei vorhergehenden/nachfolgenden Worte ist mit Tag x versehen.
- 5.) Das vorhergehende Wort ist mit dem Tag x und das nachfolgende Wort mit dem Tag y versehen.
- 6.) Das vorhergehende/nachfolgende Wort ist mit dem Tag x und das Wort zwei Schritte vor/nach dem aktuellen Wort ist mit dem Tag y versehen.

Die Templates, welche lexikalische Beziehungen berücksichtigen, lassen des Weiteren noch folgende Triggering Environments zu (Brill, 1995: 21):

„Verändere Tag a nach Tag b bei folgendem Triggering Environment:“

- 1.) Das vorangehende/nachfolgende Wort ist das Wort w.
- 2.) Das Wort zwei Schritte vor/nach dem aktuellen Wort ist das Wort w.
- 3.) Eines der zwei vorangehenden/nachfolgenden Worte ist das Wort w.
- 4.) Das aktuelle Wort ist das Wort w und das vorangehende/folgende Wort ist das Wort x.
- 5.) Das aktuelle Wort ist das Wort w und das vorangehende/folgende Wort ist mit dem Tag x versehen.

- 6.) Das aktuelle Wort ist das Wort w.
 7.) Das vorangehende/nachfolgende Wort ist das Wort w und das vorangehende/nachfolgende Tag ist das Tag x.
 8.) Das aktuelle Wort ist das Wort w, das vorangehende/nachfolgende Wort ist das Wort x and das vorangehende/nachfolgende Tag ist das Tag y.

Sowohl anhand der nicht-lexikalischen, als auch der lexikalischen Templates sollte deutlich werden, dass damit deutlich komplexere Beziehungen als mit Bi- oder Trigrammen ausgedrückt werden können.

Des Weiteren kann eine beliebige Transformation entweder einen sofortigen oder einen verzögerten Effekt haben, was sich wiederum auf nachfolgende Iterationen auswirkt. So wird aus der Kette AAAA die Kette ABAB, wenn die Transformation A->B sofort ausgeführt wird und ABBB, falls die Ausführung verzögert, also erst am Ende des Vorgangs, ausgeführt wird (Manning & Schütze, 1999: 365).

Schließlich besteht auch noch die Möglichkeit, einem Wort nicht das laut den trainierten Transformationen bestmögliche, sondern die sogenannten k-bestmöglichen Tags zuzuweisen, wodurch sich der Recall, also die korrekt erkannten Tags, erhöht, jedoch gleichzeitig die Precision, also die Rate der Worte, die ausschließlich das korrekte Tag zugewiesen bekommen haben, gesenkt wird. Um einen Kompromiss zwischen einem möglichst hohen Recall und einer möglichst hohen Precision zu erreichen, werden diejenigen Transformationen angewandt, die folgende Funktion maximieren (Brill, 1995: 31):

$$\frac{\text{Anzahl der korrigierten Fehler}}{\text{Anzahl der zusätzlichen Tags}} \quad \begin{array}{l} \rightarrow \text{höherer Recall} \\ \rightarrow \text{niedrigere Precision} \end{array}$$

Die Genauigkeit des Verfahrens liegt, falls keinerlei unbekannte Worte im Korpus vorkommen, bei 96,7% (mit lexikalischen Transformationen, 64 Kilobyte großes Trainingskorpus), 97,2 % (mit lexikalischen Transformationen, 600 Kilobyte großes Trainingskorpus) und 97,0% (ohne lexikalische Transformationen, 600 Kilobyte großes Trainingskorpus). Lexikalische Transformationen scheinen also einen eher geringen Einfluss auf die Genauigkeit des Taggers zu haben (Brill, 1995: 24).

Trotz der hohen erreichten Genauigkeit hat das Verfahren den gravierenden Nachteil, dass die Menge der möglichen Transformation, welche auf fehlerkorrigierende Transformationen durchsucht werden muss, äußerst groß ist und es daher eines effizienten Lernalgorithmus bedarf (Manning & Schütze, 1999: 366). Und selbst ein effizienter Algorithmus kann ab einer gewissen Korpusgröße nicht mehr praktikabel verwendbar sein (siehe Abschnitt 3.2).

Wenn das Modell jedoch einmal trainiert ist, erlangt ein transformationsbasierter Ansatz teilweise deutliche höhere Geschwindigkeiten beim Tagging als vergleichbare Markov Modelle (Brill, 1995: 31)

2.5 Tagging mittels Maximum Entropy Model

Ebenfalls zu den statistischen Verfahren zum Part-of-Speech Tagging zu zählen ist die Anwendung des sogenannten Maximum Entropy Model. Dieses Modell findet in der maschinellen Sprachverarbeitung vielfältige Anwendung, z.B. in der maschinellen Übersetzung (Berger et al., 1996) oder dem Erkennen von Satzgrenzen (Reyna & Ratnaparkhi, 1997), eignet sich aber insbesondere auch als zugrundeliegendes Modell beim Tagging. Es basiert auf dem Prinzip der Maximierung der Entropie von Informationen. Als Entropie wird die durchschnittliche Menge an Informationen bezeichnet, die benötigt wird, um eine bestimmte Nachricht zu übertragen (Manning & Schütze, 1999: 61).

Der im Nachfolgenden skizzierte Ansatz von Ratnaparkhi (1996) beschreibt ein solches Modell, welches vor dem Hintergrund einer Menge $H \times T$, also dem kartesischen Produkt aus H und T , definiert wird. H sei dabei die Menge der möglichen Wort- und Tagkontexte für ein Wort, also gewissermaßen der *Geschichten* (*Histories*) eines Wortes und T die Menge der zulässigen Tags. Das zugrundeliegende Wahrscheinlichkeitsmodell wird auch hier aus einem manuell getaggten Trainingskorpus automatisch erschlossen.

Die Wahrscheinlichkeit eines Tags t und einer Geschichte h ergibt sich unter diesen Voraussetzungen wie folgt (Ratnaparkhi, 1995: 1):

$$p(h,t) = \frac{\pi \mu \prod_{j=1}^k \alpha_j^{f_j(h,t)}}{\sum_{h,t} \pi \mu \prod_{j=1}^k \alpha_j^{f_j(h,t)}}$$

π steht hier für eine Normalisierungskonstante, $\{\mu, \alpha_1, \dots, \alpha_k\}$ sind die positiven Modellparameter, $\{f_1, \dots, f_k\}$ stehen für sogenannte *Features* bzw. Eigenschaften, wobei ein Wert von $f_j(h, t) \in \{0, 1\}$, also eine Eigenschaft für eine Geschichte und ein Tag entweder wahr oder falsch sein kann. Weiterhin bezieht sich jeder Parameter α_j auf eine Eigenschaft f_j . Bei einer gegebenen Wortsequenz $\{w_1, \dots, w_n\}$ und einer ebenfalls gegebenen Tagsequenz $\{t_1, \dots, t_n\}$ wird h_i als die Geschichte $\{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$ der Wort- und Tagkontexte des gerade behandelten Wortes definiert (Ratnaparkhi, 1995: 1-2).

Die Modellparameter $\{\mu, \alpha_1, \dots, \alpha_k\}$ werden daraufhin derart gewählt, dass sie die Wahrscheinlichkeit $L(p)$ der Trainingsdaten maximieren (Ratnaparkhi, 1995: 2):

$$L(p) = \prod_{i=1}^n p(h_i, t_i) = \prod_{i=1}^n \frac{\pi \mu \prod_{j=1}^k \alpha_j^{f_j(h_i, t_i)}}{\sum_{h,t} \pi \mu \prod_{j=1}^k \alpha_j^{f_j(h,t)}}$$

Eine Gesamtwahrscheinlichkeit $p(h_i, t_i)$ wird also von denjenigen Parametern α_j bestimmt, für welche die Eigenschaft gilt: $f_j(h_i, t_i) = 1$. Gegeben sei z.B. folgende Eigenschaftsfunktion $f_i(h_i, t_i)$ (Ratnaparkhi, 1995: 2):

$$\begin{aligned} f_i(h_i, t_i) &= 1, \text{ falls das } w_i = \text{'ing'} \text{ endet \& } t_i = \text{VBG (Verb, Gerundium}^7) \\ &= 0, \text{ andernfalls} \end{aligned}$$

Dies bedeutet, dass der dieser Eigenschaft entsprechende Parameter α_j aktiviert wird, falls ein Wort w_i auf 'ing' endet und das Tag VBG zugewiesen bekommen hat, also α_j mit in die Berechnung der Gesamtwahrscheinlichkeit einbezogen wird, da $\alpha_j^{f_i(h_i, t_i)} = \alpha_j^1 = \alpha_j$. Sind diese Bedingungen umgekehrt nicht gegeben so erhält die als Exponent in $\alpha_j^{f_i(h_i, t_i)}$ fungierende Eigenschaftsfunktion den Wert 0, was zu einem Resultat $\alpha_j^0 = 1$ führt, welches wiederum das Ergebnis des Produktes aus allen Parametern $\{\alpha_1, \dots, \alpha_k\}$ nicht verändert.

Wenn man nun bedenkt, dass eine Geschichte h_i als die Menge der Eigenschaften $\{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$ definiert ist und dass die verschiedenen möglichen Tags für t_i mit allen Elementen dieser Menge kombiniert werden, dürfte deutlich werden, dass die Menge H der möglichen Geschichten enorm groß ist. Aus diesem Grund wird eine bei Ratnaparkhi (1996: 2) beschriebene Näherungsfunktion zur Berechnung des Modells benutzt.

Desweiteren wird bei der Suche der wahrscheinlichsten Tagsequenz für einen Satz wie schon beim T'n'T Ansatz (siehe 2.2.3) ein Beam Search Algorithmus benutzt, um diese zu beschleunigen. Zudem verwendet die Suche optional ein sogenanntes *Tag Dictionary*, welches für jedes bekannte Wort aus dem Trainingskorpus die Tags enthält, die dieses Wort darin angenommen hat. Wird dieses Dictionary verwandt, findet die Suche nur solche Tags für ein bekanntes Wort, die auch für dieses Wort im Dictionary vorkommen. Für unbekannte Worte werden hingegen alle Tags aus dem Tagset ausgewählt. Letzteres Vorgehen findet bei nicht verwendetem Tag Dictionary für alle Worte Anwendung (Ratnaparkhi, 1995: 4).

In der Grundaustattung (*Baseline*⁸ Konfiguration) erreicht das Verfahren eine Genauigkeit von 96,43% unter Anwendung des Tag Dictionary und eine von 96,31% ohne Tag Dictionary (Ratnaparkhi, 1995: 5). Bei Verfeinerung des Modells, in deren Zuge weitere, wortspezifische Eigenschaften hinzugefügt werden, die vom grundlegenden Modell nicht erfasst werden (Ratnaparkhi, 1995: 4-6), ergibt sich eine Genauigkeit von 97,13% (Ratnaparkhi, 1995: 7).

Laut Ratnaparkhi (1996) hat das Maximum Entropy Model weiteren gängigen Verfahren gegenüber zumindest jeweils einen Vorteil. So erlaube es weitaus differenziertere linguistische Informationen zu modellieren als Markov Modelle (was jedoch, wie in den Abschnitten 2.2 und 3.1.2 über das T'n'T Verfahren deutlich werden sollte, nur bedingt für

7 Siehe Jurafsky & Martin (2005: 11)

8 Als Baseline wird die Effizienz des einfachst möglichen Algorithmus für ein Verfahren beschrieben (siehe Manning & Schütze, 1999: 234)

elaboriertere Ausprägungen dieser Modelle zutrifft) und kann, anders als transformationsbasierte Ansätze wie der von Brill (1995) im Rahmen von Wahrscheinlichkeitsmodellen eingesetzt werden.

3. Die vorgestellten Ansätze im Vergleich

3.1 Verarbeitung unbekannter Worte

3.1.1 (Hidden) Markov Modelle

In reinen HMM bzw. im Viterbi-Algorithmus wird die Verarbeitung unbekannter Worte, d.h. solcher Token, die der Tagger bisher nicht in der Lage war zu trainieren, nicht explizit berücksichtigt.

Dennoch bestehen einige Methoden zur Behandlung von Worten im Rahmen von HMM, für die keine statistischen Daten ermittelt werden konnten. Zwei dieser Ansätze sind Jelineks und Kupiecs Methode, welche beide ein Lexikon mit den möglichen Wortarten für bestimmte Worte voraussetzen.

Bei Jelineks Methode werden statistische Daten von Worten, die bisher nicht trainiert wurden, ergänzt, indem davon ausgegangen wird, dass Worte mit der gleichen Wahrscheinlichkeit von jedem ihrer möglichen Tags emittiert werden (Manning & Schütze, 1999: 357-358):

$$b_{j,l} = (b'_{j,l}C(w^l)) / (\sum w^m b'_{j,m}C(w^m))$$

$b_{j,l}$ steht dabei für die Wahrscheinlichkeit, dass ein Wort l von einem Tag j emittiert wird, $\sum w^m$ repräsentiert die Summe aller Wörter im Lexikon und $b'_{j,l}$ nimmt dabei den Wert 0 an, falls das Tag j keine Wortart repräsentiert, die für das Wort l erlaubt ist, und wird andernfalls gleich $1/\text{Anzahl der für das Wort } l \text{ erlaubten Tags}$ gesetzt.

Kupiecs Methode (Manning & Schütze, 1999: 358) funktioniert ähnlich, jedoch wird nicht für jedes Wort l ein eigener Satz an Parametern benötigt, sondern Äquivalenzklassen u_i gebildet:

$$u_i = \{w^l | j \in L \leftrightarrow \text{Tag } j \text{ ist ein mögliches Tag für Wort } l\},$$

$$L \text{ ist Teilmenge von } \{1, \dots, T\},$$

$$T \text{ ist die Anzahl verschiedener Tags im Tagset}$$

Wenn z.B. das Tag NN den Index 5 und NE den Index 8 hat, dann enthält die Äquivalenzklasse $u_{\{5,8\}}$ alle Worte, für die diese Tags möglich sind. Die obige Formel ändert sich entsprechend:

$$b_{j,L} = (b'_{j,L}C(u_l)) / (\sum_{u_r} b'_{j,L}C(u_r))$$

Auch wenn diese beide Ansätze eine Möglichkeit darstellen, nicht trainierte statistische Daten zu ergänzen, so bleibt doch anzumerken, dass gänzlich unbekannte, also nicht im Lexikon vorkommende Worte nicht von diesen erfasst werden. Daher eignen sich diese Verfahren zum Initialisieren eines Modells bei Nichtvorhandensein ausreichender Trainingsdaten, jedoch nicht zur Behandlung unbekannter Worte im eigentlichen Sinne. Hier bedarf es weitergehender Heuristiken, wie sie z.B. im folgenden Abschnitt für den T'n'T Tagger beschrieben werden.

3.1.2 T'n'T

In Brants (2000) wird die detaillierte Behandlung unbekannter Worte als ausdrückliche Anforderung für einen Tagger formuliert, welcher gute Ergebnisse erzielen soll. Als Ansatz wird hier eine *Suffixanalyse* verwandt, die Tagwahrscheinlichkeiten entsprechend ihrer Wortendungen festlegt. 'Suffix' im Sinne des hier beschriebenen Verfahrens bedeutet keine morphologische Entität, sondern schlicht 'Zeichen am Wortende'. Die Wahrscheinlichkeitsverteilung für Tags für unbekannte Worte wird dabei errechnet, indem auf die Verteilung für Worte zurückgegriffen wird, welche das gleiche *Suffix* haben. Es wird also die Wahrscheinlichkeit eines Tags t gegeben ein Suffix der Länge m für ein Wort der Länge n ermittelt:

$$P(t|l_{n-m+1}, \dots, l_n)$$

Diese Wahrscheinlichkeit wird mittels *Successive Abstraction* geglättet. Bei dieser Methode werden nach und nach Zeichen am linken Rand des Suffixes ausgelassen und so Wahrscheinlichkeiten für allgemeinere Formen des behandelten Suffixes berechnet (z.B. zuerst für 'ichen', dann für 'chen', 'hen' usw.). Diese Wahrscheinlichkeiten werden rekursiv zu einem Gesamtwert aufaddiert:

$$\begin{aligned} &P(t|l_{n-i+1}, \dots, l_n) \\ &= (P(t|l_{n-i+1}, \dots, l_n) + \theta_i P(t|l_{n-i}, \dots, l_n)) / (1 + \theta_i) \end{aligned}$$

Diese Formel wird für jedes $i = m \dots 0$ ausgeführt, θ_i stellt einen Gewichtungsfaktor dar. Die Wahrscheinlichkeitswerte für ein Tag unter der Voraussetzung eines Suffixes einer bestimmten Länge ergeben sich aus der Anzahl der Worte, welche das gegebene Suffix und das entsprechende Tag vorweisen, dividiert durch die Anzahl der insgesamt mit dem gegebenen Suffix im Korpus auftretenden Worte:

$$P(t|l_{n-i+1}, \dots, l_n) \\ = C(t|l_{n-i+1}, \dots, l_n) / C(l_{n-i+1}, \dots, l_n)$$

Die Berechnung dieser Wahrscheinlichkeiten für ein Tag gegeben ein bestimmtes Suffix funktioniert analog zur Berechnung von Wahrscheinlichkeiten für Tags gegeben ein bestimmtes Wort. Hier liegt also wiederum ein Anwendungsfall für ein Markov Modell vor. Weil dabei aber nun die Suffixe die von den Tags emittierten Zustände darstellen, wird die Wahrscheinlichkeit für ein Suffix gegeben ein bestimmtes Tag benötigt:

$$P(l_{n-i+1}, \dots, l_n | t)$$

Diese inverse Wahrscheinlichkeit lässt sich mittels *Bayes' Theorem* herausfinden (Manning & Schütze, 1999: 43):

$$P(B|A) = (P(A|B)P(B)) / P(A)$$

Brants (2000: 3) stellt nun weitere Überlegungen an, wie die verschiedenen Parameter dieser Suffixanalyse zu wählen sind, um ein möglichst gutes Ergebnis zu erzielen.

Zunächst ist der Wert für m , also die maximale Suffixlänge zu wählen. Durch empirische Untersuchungen fand Brants heraus, dass der ideale Wert für m die Länge des längsten im Korpus vorkommenden Suffixes ist, solange diese den Wert 10 nicht überschreitet.

Des Weiteren wird, wie schon bei den Gewichtungen für die Tri-, Bi- und Unigramwahrscheinlichkeiten, ein kontextunabhängiger Ansatz für die Ermittlung des Gewichtungsfaktors θ_i verwandt, d.h alle θ_i werden auf denselben Wert gesetzt. Dieser Wert ergibt sich aus der *Standardabweichung* von der *statistischen Normalverteilung* der unbedingten Wahrscheinlichkeiten für Tags im Trainingskorpus:

$$\theta_i = (1/(s-1)) \sum_{j=1}^s (P(t_j) - \bar{P})^2$$

Diese Berechnung wird für alle $i = 0 \dots m-1$ durchgeführt und der Wert s ist die Länge des Tagsets. Die durchschnittliche Wahrscheinlichkeit ergibt sich folgendermaßen und resultiert laut Brants (2000: 3) in einen Gewichtungsfaktor θ_i von 0.03 bis 0.1:

$$\bar{P} = (1/s) \sum_{j=1}^s P(t_j)$$

Außerdem werden Suffixe, analog zur Verfahrensweise bei den Tags, für groß geschriebene Worte separat von denen für klein geschriebene Worte behandelt, was das Taggingergebnis für unbekannte Worte zusätzlich verbessern soll.

Schließlich wird beim Tagging unbekannter Worte noch die Tatsache miteinbezogen, dass unbekannte Worte höchstwahrscheinlich seltene Worte sind, weswegen bei der Suffixanalyse ausschließlich Suffixe verwandt werden, die in Worten vorkommen, deren Frequenz im Korpus unter einem bestimmten Schwellenwert liegt, welcher, wiederum empirisch begründet, auf den Wert 10 festgelegt wird (Brants, 2000: 3).

Beim Tagging des NEGRA Korpus (Brants et al., 1999) des Deutschen ergibt sich beim T'n'T Verfahren bei einer Rate von 11,9% unbekannter Worte eine Genauigkeit von 97,7% bei bekannten, 89,0 % bei unbekanntem Worten und eine von 96,7% insgesamt, was ein sehr gutes Ergebnis, auch verglichen mit anderen Verfahren darstellt (Brants, 2000).

3.1.3 Constraint-Grammar-basierte Taggingverfahren

Um jegliche Eingaben, auch solche, welche unbekannte, also nicht im Lexikon vorhandene, Worte enthalten, verarbeiten zu können, muss eine Constraint Grammar über eine Strategie verfügen, auch diesen Worten Lesarten zuzuweisen, um sie bei der Disambiguierung berücksichtigen zu können. ENGTWOL, das Modul, welches für die morphologische Analyse im Rahmen der ENGCG zuständig ist bzw. das Lexikon bereitstellt, verfolgt dabei eine dreigeteilte Strategie.

Zunächst wird jedes Wort und die dafür möglichen Lesarten im Lexikon gesucht. Diesem Lexikon werden bei jedem Durchlauf die unbekanntem Worte aus dem gerade behandeltem Korpus hinzugefügt, so dass das Lexikon nach und nach erweitert wird.

Sollte ein Wort nicht gefunden werden, wird zunächst auf heuristische Methoden zurückgegriffen, welche anhand der Wortsuffixe 'raten', zu welchen Wortklassen ein bestimmtes Wort gehören könnte. So ist z.B. ein englisches Wort, welches das Suffix '-able' enthält, mit hoher Wahrscheinlichkeit ein Adjektiv.

Ist auch diese Methode nicht erfolgreich werden dem Wort schlicht alle offenen Wortklassen zugewiesen (Tapanainen & Voutilainen, 1994).

Diese Strategie hat zur Folge, dass jedes Wort analysiert werden kann und aufgrund der Weise, wie Constraints arbeiten auch die Möglichkeit besteht, dass selbst ein eigentlich unbekanntes Wort noch korrekt disambiguiert wird (siehe Constraints vom Typ =!! in Abschnitt 2.3).

3.1.4 Transformationsbasiertes Tagging

Beim Tagging mittels Transformation-Based Error-Driven Learning werden für den Tagger bisher unbekannte Worte ebenfalls mit Hilfe von speziellen Transformationen verarbeitet, die das Taggingergebnis für unbekannte Worte aufgrund morphologischer und distributioneller Eigenschaften derselben verbessern.

Unbekannte Worte werden zunächst ganz naiv als normale Substantive markiert, wenn sie klein geschrieben sind. Sie werden hingegen als Eigennamen bezeichnet, wenn sie groß geschrieben werden, womit zumindest für solche Sprachen, in denen normale Substantive klein geschrieben werden, eine relevante Entscheidung getroffen wird. Auf die so initialisierten Wort werden daraufhin alle durch die folgenden Templates lizenzierten Transformationen angewandt und auf die Verbesserung der Fehlerrate hin überprüft (Brill, 1995: 25-26):

„Verändere Tag a nach Tag b eines unbekanntes Wortes bei folgendem Triggering Environment:“

- 1.) Die Löschung des Präfixes/Suffixes x ($|x| \leq 4$) ergibt ein Wort.
- 2.) Die ersten/letzten 1...4 Zeichen des aktuellen Wortes ergeben die Zeichenkette x .
- 3.) Das Hinzufügen der Zeichenkette x ($|x| \leq 4$) als Präfix/Suffix ergibt ein Wort.
- 4.) Das Wort w taucht zumindest einmal links/rechts vom aktuellen Wort auf.
- 5.) Das Zeichen z taucht im aktuellen Wort auf.

Wie auch schon beim T'n'T Tagger (Abschnitt 3.1.2) bedeutet auch hier die Begrifflichkeit Präfix/Suffix keine unbedingt linguistische Entität, sondern schlicht eine Zeichenkette am Anfang bzw. am Ende eines Wortes, die auf keinerlei morphologischen Gesichtspunkten begründet sein muss (Brill, 1995: 26).

Nach Anwendung speziell auf unbekannte Worte zugeschnittener Transformationen können auf das Ergebnis wiederum normale kontextuelle Transformation angewandt werden, um das Ergebnis weiter zu verbessern (Brill, 1995: 28).

Der von Brill entwickelte Tagger erreicht eine Genauigkeit von 82,2% bei unbekanntes Worten und eine Gesamtgenauigkeit von 96,3% bis 96,6% bei einem Testkorpus, das unbekanntes Worte enthält (Brill, 1995: 28-30).

3.1.5 Tagging mittels Maximum Entropy Model

Der von Ratnaparkhi (1996) beschriebene Ansatz zum POS Tagging auf Grundlage eines Maximum Entropy Model macht sich die Hypothese zu nutze, dass seltene Worte - als solche werden hier Worte mit einer Frequenz im Trainingskorpus kleiner 5 definiert – in ihren Eigenschaften unbekannt, also im Trainingskorpus nicht vorkommenden, Worten ähnlich sind. Erlaubte Eigenschaftskombinationen für ein unbekanntes oder seltenes Wort w_i sind zusätzlich zu denen für nicht seltenen Worten, d.h. die in der Menge $h_i = \{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$ enthaltenen Elemente, noch alle Prä- und Suffixe, deren Länge kleiner oder gleich 4 ist (Suffix bedeutet hier wiederum wie schon in 3.1.2 und 3.1.4 keine konkrete linguistische Entität, sondern eine willkürliche wortinitiale oder –finale Zeichenkette), sowie Eigenschaften die anzeigen, ob ein Wort eine Ziffer, einen Bindestrich oder einen Großbuchstaben enthält (Ratnaparkhi, 1995: 3).

Hierbei werden alle Eigenschaftsausprägungen, die seltener als 10 Mal in den Trainingsdaten vorkommen als zu unzuverlässig ignoriert (Ratnaparkhi, 1995: 2,4).

Dieser Ansatz der Behandlung von unbekannt Worten analog zu seltenen Worten anhand deren Affixeigenschaften entspricht weitestgehend dem in Abschnitt 3.1.2 beschriebenen Ansatz von Brants (2000).

3.2 Genauigkeit im Vergleich anhand des Corpus Gesproken Nederlands

Zum Vergleich der Genauigkeit der hier aufgeführten Taggingverfahren soll auf eine Studie am Corpus Gesproken Nederlands (Zavrel & Daelemans, 1999) verwiesen werden, da sich im Rahmen dieser Studie alle Verfahren auf derselben Korpus- und Tagsetgrundlage beweisen mussten, also die gleichen Startbedingungen für alle Verfahren gegeben waren. Dabei wurden die Ansätze auf folgende Weisen miteinander verglichen:

- a)** Falls der Tagger ein eigenes Tagset benutzt, wurde auf die Genauigkeit bezüglich ihres eigenen Tagsets bei einem kleinen annotierten Ausschnitt des Corpus Gesproken Nederlands (CGN), welches insgesamt ca. 9 Millionen Token beinhaltet, geprüft.
- b)** Falls der Tagger variabel im zu benutzenden Tagset ist, wurde er
 - I.** zum einen auf die Genauigkeit bei Anwendung des WOTAN-I Tagsets auf einem ca. 600.000 Worten umfassenden Teil des Eindhoven Corpus und
 - II.** zum anderen bei Verwendung des detaillierteren WOTAN-II Tagsets auf einem ca. 150.000 Worte großen Teil des Eindhoven Corpus hin getestet.
- c)** Schließlich wurde noch eine *Projektion (Mapping)* des vom Tagger benutzten Tagsets auf das CGN-eigene Tagset durchgeführt.

d) Außerdem wurde noch für die Tagger, welche ein variables Tagset als Eingabe akzeptieren, ein Vergleich für einen 2388 Worte umfassenden Ausschnitt des Corpus Gesprochen Nederlands (CGN) angestellt.

Im Folgenden werden HMM nicht ausdrücklich berücksichtigt, da diese verschiedene Verfahren in ihren unterschiedlichen Ausprägungen umfassen - so basiert z.B., wie bereits erwähnt, der T'n'T Tagger wie auch der in Zavrel & Daelemans (1999) erwähnte WOTAN Tagger ebenfalls auf HMM – jedoch in ihrer 'Standardausführung' aufgrund ihrer mangelnden Behandlung unbekannter Worte keine Berücksichtigung finden.

Das T'n'T Verfahren erlangt bei Zavrel & Daelemans (1999) Ergebnisse von **b.I)** 89,9%, **b.II)** 83,9%, **c)** 77,5% und **d)** 82,7% und liegt damit im die Genauigkeit betreffenden Vergleich der Verfahren jeweils vorne.

Der D-Tale Tagger, der hier als Vertreter der Constraint-basierten Verfahren dienen soll, erlangt bei derselben Studie eine Genauigkeit von **a)** 82,4% und **c)** 50,6%.

Der Brill Tagger, also das transformationsbasierte Verfahren, erreicht hier einen Wert von **d)** 78,2% Genauigkeit. Dieses Verfahren wurde den anderen Tests nicht unterzogen, da der Aufwand beim Training exponentiell zur Größe des Tagsets ansteigt, weswegen die Anwendung auf das Corpus Gesprochen Nederlands zu langsam abgelaufen wäre, um sie im empirischen Vergleich berücksichtigen zu können. Aus diesem Grund wurde der Brill Tagger lediglich bei dem Vergleich auf einem 2388 Worte umfassenden Ausschnitt des CGN berücksichtigt.

Das auf dem Maximum Entropy Model basierende Verfahren MXPOST erreicht Genauigkeiten von **b.I)** 86,9%, **b.II)** 81,8 %, **c)** 77,2% und **d)** 69,7%.

Abschließend folgen noch einmal alle Ergebnisse zusammengefasst in tabellarischer Form:

	a) Eigenes Tagset	b.I) WOTAN-I	b.II) WOTAN-II	c) CGN-Mapping	d) CGN
T'n'T	-	89,90%	83,90%	77,50%	82,70%
D-Tale	82,40%	-	-	50,60%	-
BRILL	-	-	-	-	78,20%
MXPOST	-	86,90%	81,80%	77,20%	69,70%

Das Verfahren mit der allgemein höchsten Genauigkeit ist also T'n'T, gefolgt von der auf dem Maximum Entropy Model basierenden Methode. Danach folgen noch das Constraint Grammar und das transformationsbasierte Verfahren, jedoch mit der Einschränkung, dass die für diese Verfahren vorhandenen Ergebnisse unter jeweils anderen Bedingungen (Tagset und Korpusgröße) gewonnen wurden und daher nicht eindeutig vergleichbar sind.

3.3 Laufzeiteffizienz

Unter anderem aus den Beobachtungen im vorhergehenden Abschnitt sollte klar werden, dass die vorgestellten Ansätze teilweise deutliche Unterschiede aufweisen, was ihre Laufzeiteffizienz während Training und eigentlicher Ausführung betrifft.

Generell kann gesagt werden, dass statische Verfahren schneller trainieren als transformationsbasierte Verfahren wie der Brill Tagger, was darin begründet liegt, dass der Raum der möglichen Transformationen enorm groß ist. Constraint Grammar Modelle benötigen im Gegensatz dazu zwar keinerlei maschinelle Rechenzeit für ihren 'Trainingsvorgang', jedoch sollte man die menschliche Rechenzeit, welche in ein solches Modell investiert werden muss und zudem üblicherweise nicht in demselben Maße wie maschinelle Rechenzeit vorhanden ist, nicht außer Acht lassen.

Was die Geschwindigkeit während des eigentlichen Taggingprozesses betrifft, so tun sich hier besonders das T'n'T Verfahren von Brants (2000: 4) und die Methode auf Grundlage eines Maximum Entropy Model von Ratnaparkhi (1996: 4) hervor, weil in beiden Ansätzen Verfahren gefunden wurden, die Laufzeiteffizienz erheblich zu verbessern.

3.4 Beobachtungs- und Beschreibungsadäquatheit

Neben eher praktisch bedingten Gesichtspunkten beim Vergleich verschiedener Verfahren zum POS Tagging, wie z.B. deren Genauigkeit, Behandlung unbekannter Worte oder Laufzeiteffizienz, bestehen auch eher theoretisch-linguistische Aspekte, in denen sich diese unterscheiden.

Ein solcher Aspekt, in dem Unterschiede zwischen den vorgestellten Modellen deutlich werden, ist der der Beobachtungs- und Beschreibungsadäquatheit.

Eine Grammatik gilt dann als beobachtungsadäquat, wenn sie beobachtete Phänomene für eine Sprache als korrekt lizenziert. Für das POS Tagging bedeutet dies, dass eine beobachtungsadäquate Grammatik eine beobachtete Wortsequenz und die dazugehörigen Wortarten als für die Objektsprache korrekt erkennt.

Der darüber hinausgehende Begriff der Beschreibungsadäquatheit bezeichnet solche Grammatiken, die nicht nur ein sprachliches Phänomen, sondern auch die diesem Phänomen zugrundeliegende Kompetenz des Sprechers, d.h. Regularitäten korrekt modellieren.

Eine zusätzlich noch erklärungsadäquate Grammatik lässt durch ihr Modell Rückschlüsse auf die Frage zu, warum ein sprachliches Phänomen konkret so ist wie es ist. Ein solches Modell begründet also die hinter einem Phänomen stehende Struktur.

Allen hier vorliegenden Modellen ist gemein, dass sie, immer im Rahmen ihrer jeweiligen Genauigkeit, dieselbe Beobachtungsadäquatheit vorweisen, weil sie das beobachtete Phänomen der Wortsequenz und der dazugehörigen Wortarten korrekt widerspiegeln.

Bezüglich ihrer Beschreibungsadäquatheit ist festzustellen, dass allen statistischen Verfahren (reine HMM, T'n'T, Maximum Entropy Model) gemein ist, dass sie zwar die beobachteten Phänomene korrekt repräsentieren, aber aufgrund ihrer Natur sprachliche Regelmäßigkeiten nicht mittels allgemeiner Regeln, sondern mittels ihrer statistischen Relevanz erklärt werden. Regelbasierte Modelle sind im Gegensatz hierzu, wie schon ihr Name andeutet, eher dazu geeignet, sprachliche Regelmäßigkeit auch in angemessenen Regeln zu beschreiben und sind somit näher am Ideal der Beschreibungsadäquatheit als statistische Verfahren. Nicht außer Acht gelassen werden sollte hier jedoch, dass die Kompetenz eines Sprechers nicht ausschließlich von abstrakten Regeln, sondern durchaus auch von statistischen Daten abhängt. So werden z.B. bestimmte Worte in bestimmten Kontexten oder Textarten häufiger verwandt als andere.

4. Zusammenfassung, Ausblick und Fazit

Die Problemstellung eines möglichst genauen maschinellen Part-of-Speech Taggings, also das möglichst korrekte und eindeutige automatische Zuweisen von durch sogenannte Tags repräsentierten Wortarten zu Wörtern in einem Korpus, lässt sich mittels verschiedener Ansätze angehen.

Hier wurde das Augenmerk zum einen auf statistische, dann auf regelbasierte und schließlich auf solche Verfahren gelegt, welche eine Hybridform darstellen.

Konkret wurde zunächst der allgemeine Hintergrund der sogenannten Markov Modelle, sowohl in der grundlegenden, 'sichtbaren', wie auch der 'versteckten' ('hidden') Variante, beleuchtet und der Viterbi-Algorithmus vorgestellt. Dieser Algorithmus ermöglicht es, statistische Vorhersagen über eine Sequenz von Zuständen eines Hidden Markov Modells anhand der von den Zuständen emittierten Token zu treffen.

Das T'n'T bzw. 'Trigrams'n'Tags' Verfahren von Brants (2000), stellt eine Verfeinerung der Markov Modellen zugrundeliegenden Idee bzw. eine Erweiterung des Verfahrens auf Trigramme dar. Unter Verwendung des Viterbi-Algorithmus und eines Markov Modells zweiter Ordnung nutzt das Verfahren weitere Methoden und Heuristiken, um Genauigkeit und Laufzeitverhalten zu verbessern, wie z.B. lineare Interpolation zur Glättung der statistischen Daten und einen Beam Search Algorithmus zur Erhöhung der Effizienz des Viterbi-Algorithmus.

Einen gänzlich anderen Weg als Verfahren auf Markov Modell Basis schlagen hingegen die sogenannte regelbasierten Verfahren ein.

Hier wurde zunächst der von Karlsson (1995) und Voutilainen (1995) beschriebene Ansatz einer Constraint Grammar für das Englische (ENGCG) skizziert. Im Unterschied zu den statistischen Methoden trainiert dieses Verfahren das ihm zugrundeliegende Modell nicht automatisch, sondern bekommt ein manuell kreierte grammatisches Regelwerk als Eingabe vorgegeben.

Ein weiterer regelbasierter Ansatz ist das Tagging auf Basis von sogenannten Transformationen, wie es von Brill (1995) beschrieben wurde. Dieses Verfahren nutzt zwar wie der Constraint Grammar Ansatz ebenfalls Regeln zur syntaktischen Disambiguierung von Wortklassen, jedoch trainiert es diese Regeln innerhalb eines durch bestimmte Schablonen vorgegebenen Rahmens.

Schließlich wurde hier noch ein weiteres statistisches Verfahren zum POS Tagging vorgestellt, das sich jedoch stark von den auf Markov Modellen beruhenden Verfahren unterscheidet. Dieses Verfahren von Ratnaparkhi (1996) nutzt ein Maximum Entropy Model und basiert auf dem Prinzip der Maximierung der Entropie von, in diesem Fall sprachlichen, Informationen. Dabei wird die Gesamtwahrscheinlichkeit eines Tokens gegeben seiner Geschichte, also einer definierten Sequenz von vorhergehenden Worten und entsprechenden Tags, berechnet.

Im folgenden Vergleich der Verfahren wurde besonderer Wert auf eine Beschreibung der jeweiligen Methodik zur Verarbeitung unbekannter, also nicht bereits im Lexikon des jeweiligen Taggers existenter Worte, gelegt.

So nutzt der T'n'T Tagger eine elaborierte Suffixanalyse, welche der Hypothese Rechnung trägt, dass unbekannte Worte höchstwahrscheinlich selten sind und seltene Worte daher in Distribution, Wortarten und wortinterner Struktur unbekanntem Worten zumindest ähneln.

Im Rahmen der Constraint Grammar Methodik wird versucht, die möglichen Wortarten eines unbekanntem Wortes mittels einer morphologischen Analyse zu 'raten'. Falls dieses Verfahren erfolglos ist, werden einem unbekanntem Wort hier schlicht alle möglichen Wortarten zugeordnet, damit alle Worte für den Tagger zumindest verarbeitbar sind. Zudem besteht aufgrund der Natur einer Constraint Grammar selbst dann noch die Möglichkeit einer erfolgreichen Disambiguierung.

Beim transformationsbasierten Ansatz werden spezielle Arten von Auslösungsbedingungen (Trigger) für die Verarbeitung unbekannter Worte definiert. Diese erinnern in ihrer Grundidee dem Ansatz des T'n'T Taggers, da hier vornehmlich auf wortinitiale und -finale Zeichenketten Wert gelegt wird.

Derselbe Gedanke findet sich ebenfalls beim auf dem Maximum Entropy Model basierenden Verfahren wieder, denn auch hier werden bei unbekanntem Worten zusätzlich zu den Geschichten eines Wortes, die wortfinalen Zeichenketten eines Wortes, sowie verschiedene orthographische Konventionen (Bindestrich, Großbuchstaben, wortinterne Ziffern) berücksichtigt. Ebenso wird auch hier der Ansatz verfolgt, dass seltene Worte unbekanntem Worten in Distribution, Wortarten und wortinterner Struktur ähneln.

Es lässt sich also festhalten, dass alle hier vorgestellten Verfahren in irgendeiner Weise Affixanalyse betreiben, um die Wortarten unbekannter Worte zu erraten. Außerdem sticht die offensichtliche Relevanz der Hypothese hervor, dass unbekannte Worte prinzipiell selten vorkommende Worte sind.

Dies findet sich z.B. auch in einer Analyse von Jurafsky & Martin (2005: 41) wieder, in der festgestellt wird, dass sowohl unbekannte Worte als auch *Hapax Legomena*, also nur einmal in einem Korpus vorkommende Worte, am wahrscheinlichsten Substantive, gefolgt von Verben, jedoch höchstwahrscheinlich keine Artikel oder Interjektionen sind.

Ein weiterer Ansatz zur Verbesserung der Tagginggenauigkeit bei unbekanntem Worten ist die sogenannte *Named Entity Recognition*. Darunter fällt z.B. ein *Company Name Detector*, der erkennt, ob ein Wort bestimmte auf einen Firmennamen und somit auf ein als Eigennamen zu taggendes Wort hindeutende Zeichenketten wie z.B. 'Ltd.', 'PLC' für das Englische oder 'GmbH', 'AG' etc. für das Deutsche enthält (Jurafsky & Martin, 2005: 43).

Schließlich wurde noch die Genauigkeit der verschiedenen Verfahren anhand einer Studie über die Effizienz verschiedener Taggingverfahren bei Anwendung auf das Corpus Gesprochenen Nederlands (Zavrel & Daelemans, 1999) verglichen.

Außerdem folgten noch einige Anmerkungen zur Laufzeiteffizienz und der Beobachtungs- und Beschreibungsadäquatheit der verglichenen Verfahren, wobei deutlich gemacht werden sollte, dass die einzelnen Verfahren nicht immer unbedingt die Regelmäßigkeiten der sprachlichen Wirklichkeit repräsentieren.

Als möglicher Ausblick sei hier noch ein weitergehender Ansatz genannt, die Genauigkeit beim automatischen POS Tagging zu verbessern. Dieser wird von Zavrel & Daelemans (1999: 8) als 'synergie van tagger-combinatie' und von Brants (2000: 1) als 'voting tagger' bezeichnet. Gemeint ist damit ein Framework, in dem verschiedene Tagger die Ergebnisse untereinander verbessern, indem das Tag als korrekt für ein Wort erkannt wird, für das die meisten Tagger 'gestimmt' haben.

Zavrel & Daelemans (1999: 8) weisen in diesem Zusammenhang darauf hin, dass die von unterschiedlichen Taggingmethoden jeweils falsch erkannten Worte bis zu einem gewissen Maße nicht übereinstimmen, wodurch sich der Ansatz anbietet, dass sich verschiedene Verfahren gegenseitig korrigieren. Durch ein solches Vorgehen ließe sich die Fehlerrate beim Tagging um 10 bis 20% korrigieren. Zudem seien die Worte, bei deren Wortart sich die unterschiedlichen Taggingarchitekturen nicht einig seien, häufig genau diejenigen, welche auch besonderer Aufmerksamkeit bei der manuellen Annotation bedürfen.

5. Literatur

- Berger, Adam/Della Pietra, Stephen A./Della Pietra, Vincent J. (1996): *A Maximum Entropy Approach to Natural Language Processing*. In: *Computational Linguistics 1996: Volume 22, Issue 1*; 39-71. Cambridge, MA: MIT Press.
- Brants, Thorsten (2000): *TnT - A Statistical Part-of-Speech Tagger*. In: *Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000*. Seattle, WA.
- Brants, Thorsten/Skut, Wojciech/Uszkoreit, Hans (1999): *Syntactic annotation of a German newspaper corpus*. In: *Proceedings of the ATALA Treebank Workshop*, 69-76. Paris, Frankreich.
- Brill, Eric (1995): *Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging*. In: *Computational Linguistics 1995: Volume 21, Issue 4*; 543-565. Cambridge, MA: MIT Press.
- Halama, André (2004), *Flache Satzverarbeitung*. In: Klabunde, R. et al. (Hrsg.): *Computerlinguistik und Sprachtechnologie. Eine Einführung*, 218-230. Heidelberg: Elsevier.
- Jurafsky, Daniel/Martin, James H. (2005): *Speech and Language Processing. Chapter 4*, Draft of July 7, 2005: <http://www.cs.colorado.edu/~martin/SLP/Updates/ch4.pdf> (mit freundlicher Genehmigung der Autoren).
- Karlsson, F. (1995): *The formalism and environment of Constraint Grammar Parsing*. In: Karlsson, F., Voutilainen, A., Heikkilä, J., Anttila, A. (Hrsg.): *Constraint Grammar*; 165-285. Mouton de Gruyter: Berlin.
- Kempe, André (1997): *Finite-state transducers approximating hidden markov models*. In *Proc. 35th Annual Meeting of the Association for Computational Linguistics (ACL'97)*, 460-467, Madrid, Spanien.
- Manning, Christopher D./Schütze, Hinrich (1999): *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.
- Ratnaparkhi, Adwait (1996): *A maximum entropy model for part-of-speech tagging*. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing EMNLP-96*. Philadelphia, PA.
- Reynar, J. C./Ratnaparkhi, A. (1997): *A Maximum Entropy Approach to Identifying Sentence Boundaries*. In: *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 16-19, Washington, DC.
- Schiller, Anne/Teufel, Simone/Stöckert, Christine/Thielen, Christine (1999): *Guidelines für das Tagging deutscher Textcorpora mit STTS (Kleines und großes Tagset)*. Stuttgart: Universität Stuttgart, Institut für maschinelle Sprachverarbeitung u. Tübingen: Universität Tübingen, Seminar für Sprachwissenschaft.
- Tapanainen, Pasi/Voutilainen, Atro (1994): *Tagging accurately - Don't guess if you know*. In *ANLP 1994*: 47-52. Stuttgart.
- Voutilainen, A. (1995): *Morphological Disambiguation*. In: Karlsson, F., Voutilainen, A., Heikkilä, J., Anttila, A. (Hrsg.): *Constraint Grammar*; 165-285. Mouton de Gruyter: Berlin.
- Zavrel, Jakob/Daelemans, Walter (1999): *Evaluatie van part-of-speech taggers voor het corpus gesproken nederlands*. Tilburg: CGN technical report, Katholieke Universiteit Brabant, Niederlande.