Der Viterbi-Algorithmus.

Eine Erläuterung der formalen Spezifikation am Beispiel des Part-of-Speech Tagging.

Kursskript

Karin Haenelt, 19.10.2007 ([1]11.05.2002)

1 Einleitung

In diesem Skript wird der Viterbi-Algorithmus erläutert. Der Viterbi-Algorithmus dient dazu, auf effiziente Weise die beste Pfadsequenz durch ein Hidden Markov Model zu finden.
Als Beispielanwendung wird hier das Part-of-Speech Tagging, also die Auszeichnung von Wörtern des Textes mit Wortarten, gewählt. Die formale Spezifikation des Algorithmus wird in der Version von Manning und Schütze (2000) gegeben. In der Erläuterung wird auch die Darstellung von Allen (1995) verwendet.

Abschnitt 2 gibt eine kurze Einführung in Hidden Markov Models, Abschnitt 3 führt das gewählte Beispiel ein und gibt die Werte an, die für die Verarbeitung des Beispiels verwendet werden. Abschnitt 4 gibt eine allgemeine Erläuterung der Funktionsweise des Viterbi-Algorithmus. Abschnitt 5 stellt die Notationskonventionen und Definitionen zusammen, die in der formalen Spezifikation verwendet werden. In Abschnitt 6 folgt die formale Spezifikation. In Abschnitt 7 wird die Verarbeitung des Beispiels auf drei Arten verfolgt: durch Tracing der Verarbeitungsschritte, durch schrittweise Verfolgung der für die Zwischenergebnisse aufgebauten Tabellen (Gitter / trellis / Arrays) und durch graphische Repräsentation der erreichten Verarbeitungszustände.

2 Hidden Markov Model

Ein Hidden Markov Model (HMM) ist ein stochastischer Prozess, der auf einer Markow-Kette beruht.

Definition 1 *Stochastischer Prozess*

Ein *stochastischer Prozess* oder *Zufallsprozess* ist eine Folge von elementaren Zufallsereignissen $X_1, X_2, \ldots, X_i \in \Omega$, $i = 1, 2, \ldots,$ und Ω ist die Menge der möglichen Elementarereignisse ■

Definition 2 *Zustände eines stochastischen Prozesses*

Die möglichen Zufallswerte in einem stochastischen Prozess heißen Zustände des Prozesses. Man sagt, dass sich der Prozess zum Zeitpunkt *t* in Zustand X_t befindet. ■

Beispiel

Ein Textgenerator hat ein Lexikon mit drei Wörtern, von denen an jeder Position jedes auftreten kann (Ω = {*geschickt*, *werden*, *wir*})

Wir beobachten an jeder Position, welches Wort generiert wurde. Sei $X_1 = x_1$ das Wort (x_1)zum ersten Beobachtungszeitpunkt (X_1), $X_2 = x_2$ das Wort zum zweiten Beobachtungszeitpunkt, usw. Dann ist die Folge der Wörter ein stochastischer Prozess mit diskreter Zufallsvariable und diskretem Zeitparameter

Für die vollständige Beschreibung eines Zufallsprozesses mit diskretem Zeitparameter benötigt man

1. die Anfangswahrscheinlichkeit:
   die für jeden Zustand angibt, mit welcher Wahrscheinlichkeit er als Zustand X_1 beobachtet werden kann (d.h. den Startzustand bildet)
   $\pi_i = P(X_1 = s_i)$
2. die Übergangswahrscheinlichkeit:
   die für jeden Zustand angibt, mit welcher Wahrscheinlichkeit er in einer Zustandsfolge auftritt
   $P(X_{t+1} = x_{t+1} \mid X_1 = x_1, X_2 = x_2, \ldots, X_t = x_t)$

Definition 3 *Markow-Kette*

Eine Markow-Kette ist ein spezieller stochastischer Prozess, bei dem zu jedem Zeitpunkt die Wahrscheinlichkeiten aller zukünftigen Zustände nur vom momentanen Zustand abhängen (= Markow-Eigenschaft), d.h. es gilt $P(X_{t+1} = x_{t+1} \mid X_t = x_t)$. ■

Definition 4 *endliche Markow-Kette*

Für eine endliche Markow-Kette gibt es endlich viele Zustände, und die Kette muss sich zu jedem Zeitpunkt in einem dieser endlich vielen Zustände befinden. ■

Eine endliche Markow-Kette kann durch eine stochastische Matrix beschrieben werden.
Beispiel:

stochastische Übergangsmatrix

es gilt: $a_{ij} = P(X_{t+1} = s_j \mid X_t = s_i)$

$\forall$ i,j ≥ 0

$\forall$ i $\sum_{j=1}^{N} a_{i,j} = 1$

$X_t = s_i$	$X_{t+1} = s_j$		
	geschickt	werden	wir
geschickt	.3	.4	.3
werden	.4	.2	.4
wir	.3	.4	.3

Anfangswahrscheinlichkeiten Π

es gilt: $\pi_i = P(X_1 = s_i)$

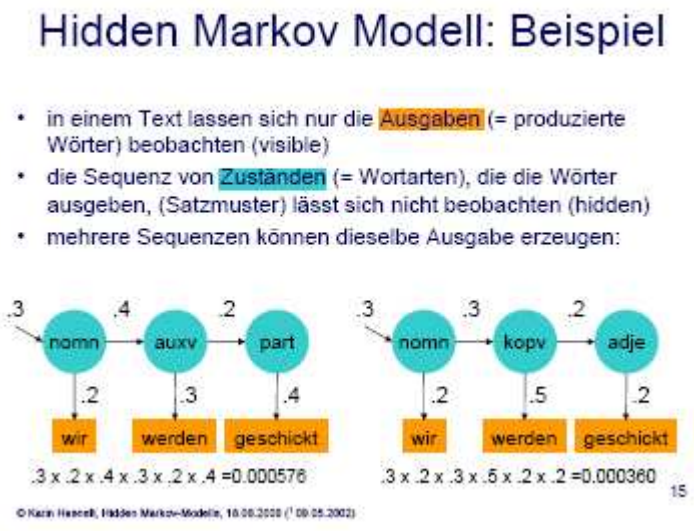
$\sum_{i=1}^{N} \pi_i = 1$

X_t	π
geschickt	.2
werden	.3
wir	.5

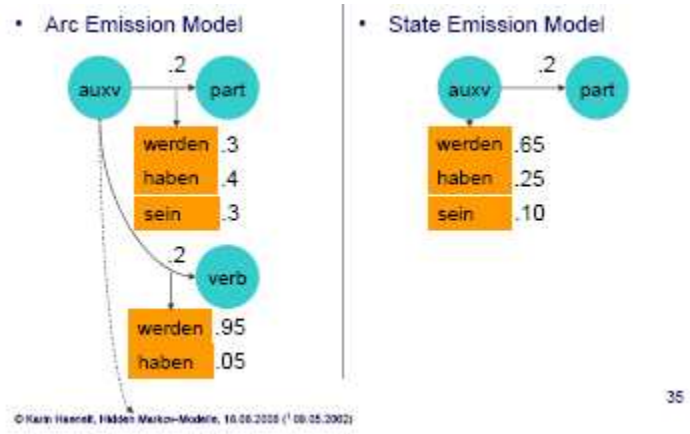
Definition 5 *Hidden Markov Model (HMM)*

Ein Hidden Markov Model ist ein Markow-Modell

- bei dem nur die Sequenz der Ausgaben beobachtbar ist,
- die Sequenz der Zustände verborgen bleibt. ■



Es gibt zwei Formen von Hidden Markov Models, nämlich *Arc Emission Models* und *State Emission Models*. Arc Emission Models geben ihre Emissionen beim Zustandsübergang aus. Sie bilden den allgemeinen Fall. State emission models geben ihre Emissionen am Zustand aus. Sie bilden einen einen Spezialfall eines arc emission models. Jedes state emission model ist in ein arc emission model überführbar, umgekehrt ist die snicht immer möglich.



Arc Emission Model:
Darstellung als Wahrscheinlichkeitsmatrix

X_t	Übergangsmatrix X_{t+1} Adje	AuxV	KopV	Nomn	Part	Punkt	Start π
Adje	.2 Emissionsmatrix o_t: geschickt .2, werden 0, wir 0, ...8	.1	.1	.4	.1	.1	.3
AuxV	.2	.3	.1	.1	.2	.1	.2
KopV	.2 Emissionsmatrix o_t: geschickt 0.05, werden .5, wir .05, ...4	.1	.1	.4	.1	.1	.1
Nomn	.05	.4	.3	.05	.1	.1	.3
Part	.3	.1	.1	.1	.3	.1	.1
Punkt	.2	.2	.1	.3	.1	.1	.1

37

Arc Emission Model:
Spezialfall: State Emission Model

X_t	Übergangsmatrix X_{t+1} Adje	AuxV
Adje	.2 Emissionsmatrix o_t: geschickt .2, werden 0, wir 0, ...8	.2 Emissionsmatrix o_t: geschickt .2, werden 0, wir 0, ...8
AuxV		
...		

Wenn die Emissionsverteilungen
für alle Übergänge aus einem Zustand
identisch sind,
entspricht dies einem State Emission Modell

38

Ein Hidden Markov Model wird spezifiziert durch die Angabe des Fünf-Tupels (S,K, Π ,A,B) (Manning/Schütze, 2000, S. 324):

- $S = \{s_1, \ldots, s_N\}$ endliche Menge von Zuständen

- $K = \{k_1,...,k_M\} = \{1,...,M\}$ endliche Menge von Ausgabe-Symbolen
- $\Pi = \{\pi_i\}, i \in S$ Menge der Wahrscheinlichkeiten der Startzustände
- $A = \{a_{ij}\}, i, j \in S$ Wahrscheinlichkeiten der Zustandsübergänge

$$\sum_{j=1}^{N} a_{ij} = 1$$

- $B = \{b_{ijk}\}, i, j \in S, k \in K$ Wahrscheinlichkeiten der Symbolemissionen

$$\sum_{k=1}^{M} b_{ijk} = 1$$

Es gilt $b_{ijk} = P(O_n = k \mid X_n = s_i, X_{n+1} = s_j)$

In dieser Spezifikation wird ein *arc-emission-Modell* angenommen, d.h. ein Modell, bei dem die Ausgabe beim Übergang von Zustand i zu Zustand j erfolgt. Beim *state-emission-Modell* ist die Ausgabe jeweils mit dem Zustand j verbunden. Die Spezifikation für die Ausgabewahrscheinlichkeit für das state-emission-Modell lautet:

$$B = \{b_{jk}\}, i, j \in S, k \in K \quad \sum_{k=1}^{M} b_{jk} = 1$$

3 Beispiel

Der Viterbi-Algorithmus sucht auf effiziente Weise die wahrscheinlichste Pfadsequenz für eine gegebene Beobachtung.

Wollen wir ein Hidden Markov Model und den Viterbi-Algorithmus für das part-of-speech-tagging verwenden, so müssen wir zuerst einmal ein Hidden Markov Model aus Beispieldaten aufbauen. Dazu benötigen wir ein Corpus, das mit part-of-speech Kategorien annotiert (getaggt ist). Wir werden hier ein ganz kleines Beispiel verwenden, und zwar wollen wir den (mehrdeutigen) Satz *„Wir werden geschickt“* nehmen. In unserem Beispiel werden wir ein state emission Modell verwenden, das allerdings leicht in ein arc emission model überführt werden kann.

Wir nehmen an, dass die Menge der Tags folgende Kategorien enthält:

Adje Adjektiv
AuxV Hilfsverb
KopV Kopulaverb
Nomn Nomen
Part Partizip II

sowie folgendes zusätzliche Symbol:

Ω Startsymbol (entspricht PERIOD bei Manning und Schütze (2000)).

Wir nehmen weiter an, dass in einem Beispielcorpus ermittelt wurde, wie oft welche Kategorie auf welche andere Kategorie folgt, und wie oft welche Kategorie welches Wort im Text klassifiziert.

Angenommen das Beispielcorpus enthält die folgenden Sätze mit den folgenden Tags:

Wir	*werden*	*geschickt*	*.*	*Wir*	*werden*	*geschickt*	*.*	*...*	*...*
nomn	auxv	part	Ω	nomn	kopv	adje	Ω	…	…

Dann können in diesem Corpus folgende Abfolgen mit folgender Häufigkeit beobachtet werden:

tag^j	tag^k und Abfolgehäufigkeiten $F(tag^k \mid tag^j)$						$word^l$ und Ausgabehäufigkeiten $F(word^l \mid tag^j)$			
	Adje	AuxV	KopV	Nomn	Part	Ω	geschickt	werden	wir	.
Adje	-	-	-	-	-	1	1	-	-	-
AuxV	-	-	-	-	1	-	-	1	-	-
KopV	1	-	-	-	-	-	1	-	-	-
Nomn	-	1	1	-	-	-	-	-	2	-
Part	-	-	-	-	-	1	-	-	-	-
Ω	-	-		1	-	-	-	-	-	2

Wenn man die Häufigkeiten nicht in absoluten Zahlen angibt, sondern in ihre prozentualen Anteile je Kategorie umrechnet, erhält man ein Hidden Markov Model für das Corpus. Für das obige Beispiel also

tag^j	tag^k und Abfolgehäufigkeiten $P(tag^k \mid tag^j)$						$word^l$ und Ausgabehäufigkeiten $P(word^l \mid tag^j)$			
	Adje	AuxV	KopV	Nomn	Part	Ω	geschickt	werden	wir	.
Adje	-	-	-	-	-	1.0	1.0	-	-	-
AuxV	-	-	-	-	1.0	-	-	1.0	-	-
KopV	1.0	-	-	-	-	-	1.0	-	-	-
Nomn	-	0.5	0.5	-	-	-	-	-	1.0	-
Part	-	-	-	-	-	1.0	-	-	-	-
Ω	-	-		1.0	-	-	-	-	-	1.0

D.h. auf ein Nomn folgt zu 50% ein AuxV und zu 50% ein KopV.

Für das nun folgende Beispiel des Part-of-Speech Tagging wollen wir annehmen, dass wir auf Grund der Auszählungen in einem Beispielcorpus das folgende state-emission-Modell gewonnen haben:

tag^j	tag^k und Übergangswahrscheinlichkeit						$word^l$ und Emissionswahrscheinlichkeit					Start-wahrscheinlichkeit
	$P(tag^k \mid tag^j)$						$P(word^l \mid tag^j)$					$\Pi\ (tag^j)$
X_t	X_{t+1}						o_t					π
	Adje	AuxV	KopV	Nomn	Part	Ω	ge-schickt	werden	wir	.	...	
Adje	.2	.1	.1	.4	.1	.1	.2	0	0	0	.8	.3
AuxV	.2	.3	.1	..1	.2	.1	0	.3	0	0	.7	.2
KopV	.2	.1	.1	.4	.1	.1	0	.5	0	0	.5	.1
Nomn	.05	.4	.3	.05	.1	.1	0	0	.2	0	.8	.3
Part	.3	.1	.1	.1	.3	.1	.4	0	0	0	.6	.1
Ω	.3	.2	.1	.3	.1	.0	0	0	0	1	0	1

In der Beispielanwendung des part-of-speech Tagging sind die Zustände (Knoten) des Automaten die Wortarten. Die Kanten zwischen den Knoten beschreiben die möglichen Zustandsübergänge und ihre Wahrscheinlichkeiten. Im Beispiel wird dadurch die Wahrscheinlichkeit der Abfolge der Wortarten beschrieben.
Zu jedem Zustand gehört außerdem eine Emissionsmatrix, die angibt, mit welcher Wahrscheinlichkeit in welchem Zustand welche Ausgabe erfolgt. Im Beispiel sind die Ausgaben der Zustände die Wörter.

Besonders zu beachten ist, dass hier angegeben wird, mit welcher Wahrscheinlichkeit z.B. ein Partizip das Wort *„geschickt“* ausgibt, und nicht etwa mit welcher Wahrscheinlichkeit das Wort *„geschickt“* ein Partizip ist. Nur die erste Betrachtungsweise erfüllt die stochastische Bedingung, dass die Summe der Wahrscheinlichkeiten der Ausgaben gleich 1 ist. Die Bedingung lautet: $\sum_{k=1}^{M} b_{ijk} = 1$, und die Definition lautet $b_{ijk} = P(O_n = k \mid X_n = s_i, X_{n+1} = s_j)$. Würde man dagegen die Wahrscheinlichkeit betrachten, mit der das Wort „*geschickt*“ z.B. ein Partizip ist, würde die Summe der Ausgaben einer Kategorie beliebig. Die Kategorien bilden

aber im Beispiel die Zustände, und die Zustände müssen die stochastischen Bedingungen erfüllen. Es werden aus der Sicht eines Zustandes zwei Dinge überprüft, nämlich erstens, wie gut er in die Abfolge der Zustände (hier: Kategorien) passt, und zweitens, wie gut er die aktuelle Beobachtung vorhersagt. Nur so sind zwei Ergebnisse auf der Basis des Modells vergleichbar.

Die Zustände, die der Automat durchläuft, können nicht beobachtet werden, sondern nur die Ausgaben. Dies entspricht der Situation im Part-of-Speech Tagging: in einem Text können nur die Wörter beobachtet werden, nicht ihre Kategorien. Die Kategorien werden durch Interpretation zugewiesen. Das Hidden Markov Model fungiert als Interpretationsmodell. Für das Part-of-Speech Tagging wird der Pfad durch den Wortart-Graphen gesucht, der nach dem gegebenen Modell μ mit der höchsten Wahrscheinlichkeit die beobachtete Kette *„Wir werden geschickt“* ausgibt; d.h. $\arg_X \max P(X|O, \mu)$ bzw. $\arg_X \max P(X|O_1=wir, O_2=werden, O_3=geschickt, \mu)$, wobei X die Kette der Zufallsvariablen ist, deren Werte Elemente der Menge S (Wortarten) sind, und O die Kette der Variablen, deren Werte Elemente der Menge K (Wörter) sind. Anders ausgedrückt: gesucht ist die beste Sequenz der nicht-beobachtbaren Zustände (Wortarten), die zu den Wörtern des Textes führen; also die Abfolge von Wortarten, die am wahrscheinlichsten zu den Wörtern eines Textes passt.

Anstatt mit einer Tabelle der Startwahrscheinlichkeiten zu arbeiten, kann man auch dem zu analysierenden Text einen Punkt (hier: Kategorie Ω) voranstellen und dann nur mit den Übergangs- und Emissionswahrscheinlichkeiten arbeiten. Die Übergänge von Kategorie Ω zu den anderen Kategorien entsprechen den Startwahrscheinlichkeiten für die anderen Kategorien. Unser Beispiel lautet nun: *„., wir, werden, geschickt“.* (so ist auch das Beispiel in Manning/Schütze 2000 aufgebaut).

4 Funktion des Viterbi-Algorithmus

Der Viterbi-Algorithmus sucht die wahrscheinlichste Sequenz der verborgenen Zustände in einem gegebenen Hidden Markov Modell zu einer gegebenen Beobachtung.
Ein sehr ineffizienter Weg, zu einer gegebenen Beobachtung O (hier: *„., wir, werden, geschickt“*) und einem gegebenen Modell μ (s. Werte in Abschnitt 2), den besten Pfad durch das Übergangsnetzwerk zu finden, wäre es,

- alle möglichen Pfade zu ermitteln,
- alle Pfade auszusuchen, die die gegebene Beobachtung ausgeben,
- hiervon den Pfad mit der höchsten Wahrscheinlichkeit auszuwählen.

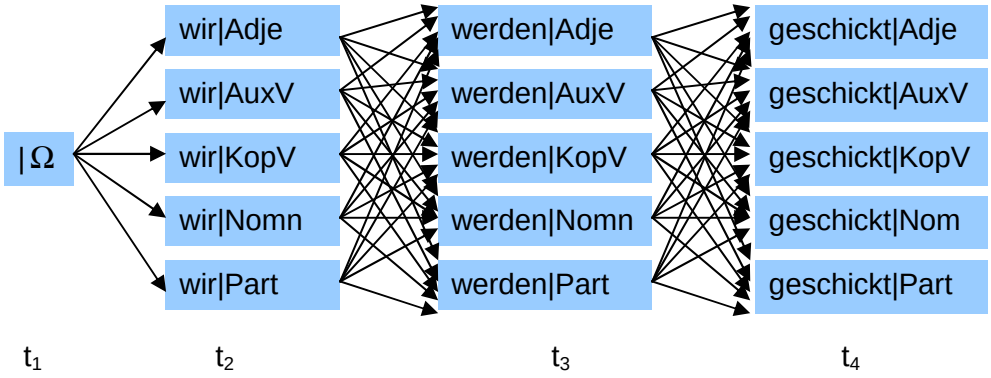
Bei 5 Zuständen und 3 Beobachtungen sind das 5^3 Pfade.

1	Ω	→	*wir* | Adje	→	*werden* | Adje	→	*geschickt* | Adje
2							*geschickt* | AuxV
3							*geschickt* | KopV
4							*geschickt* | Nom
5							*geschickt* | Part
6					*werden* | AuxV	→	*geschickt* | Adje
7							*geschickt* | AuxV
..							…
..			…		…		…
85			*wir* | Nomn	→	*werden* | AuxV	→	*geschickt* | Part
86			*wir* | Nomn	→	*werden* | KopV	→	*geschickt* | Adje
..			…				
125			*wir* | Part	→	*werden* | Part	→	*geschickt* | Part

auf denen sich folgende Werte ergeben:

P(Adje Adje Adje | *wir,werden,geschickt*, μ)		=0
P(Adje Adje AuxV | *wir,werden,geschickt*, μ)		=0
...		...
P(Nomn AuxV Part | *wir,werden,geschickt*, μ)	.3 x .2 x .4 x .3 x .2 x .4	=0.000576
P(Nomn KopV Adje | *wir,werden,geschickt*, μ)	.3 x .2 x .3 x .5 x .2 x .2	=0.000360
...		...
P(Part Part Part | *wir,werden,geschickt*, μ)		=0

Im Viterbi-Algorithmus werden diese 125 Pfade kompakt als Gitter (engl. ‚trellis') dargestellt. Wie beim Chart-Parsing werden hier partielle Zwischenergebnisse nicht immer wieder neu berechnet, sondern wiederholt weiterverwendet. Außerdem wird für jeden Zeitpunkt t nur die Wahrscheinlichkeit des wahrscheinlichsten Pfades, der zu einem Knoten führt, und der Vorgängerknoten auf diesem Pfad gespeichert.



5 Notationskonventionen und Definitionen

Zur Spezifikation des Algorithmus werden folgende Notationskonventionen benutzt (Manning/Schütze, 2000, S. 346):

w_i	Wort an Position i im Corpus
t_i	Tag von Wort w_i
w^l	das l.te Wort im Lexikon
t^j	das j.te Tag in der Tag-Menge
T	Anzahl der Tags in der Tag-Menge
W	Anzahl der Wörter im Lexikon
n	Satzlänge

6 Formale Spezifikation des Viterbi-Algorithmus

Manning und Schütze (2000, S. 350) geben folgende formale Spezifikation des Algorithmus:

1 **comment:** Given: a sentence of length n

2 **comment:** Initialization

3 $\delta_1(\Omega) = 1.0$

4 $\delta_1(t) = 0.0$ for $t \neq \Omega$

5 **comment:** Induction

6 **for** *i* := 1 **to** n **step** 1 **do**

7 **for** all tags t^j **do**

8 $\delta_{i+1}(t^j) := \max_{1 \leq k \leq T} [\delta_i(t^k) \times P(w_{i+1}|t^j) \times P(t^j|t^k)]$

9 $\psi_{i+1}(t^j)$:= arg max$_{1\le k\le T}$ [$\delta_i(t^k)$ x P($w_{i+1}|t^j$) x P($t^j|t^k$)]

10 **end**

11 **end**

12 **comment:** Termination and path-readout

13 X_{n+1} = arg max$_{1\le j\le T}$ δ_{n+1} (j)

14 **for** j := n **to** 1 **step** -1 **do**

15 $X_j = \psi_{j+1}(X_{j+1})$

16 **end**

17 P(X_1, …, X_n) = max $_{1\le j\le T}$ δ_{n+1} (t^j)

7 Erläuterung der formalen Spezifikation mit Tracing des Viterbi-Algorithmus

Die Werte des Gitters (trellis) werden in der Initialisierung und in der Induktion (von Allen (1995) Iteration genannt) aufgebaut. Zu diesem Zweck werden zu jedem Zeitpunkt t (d.h. für jedes Wort) für jeden Knoten (d.h. für jede Wortart) zwei Funktionen berechnet:

$\delta_{i+1}(t^j)$ berechnet für jeden Knoten im Gitter (trellis) die Wahrscheinlichkeit des wahrscheinlichsten Pfades, der bis zu diesem Knoten führt

$\psi_{i+1}(t^j)$ ermittelt den Knoten der hereinkommenden Kante, die zu diesem wahrscheinlichsten Pfad geführt hat, d.h. den Vorgängerknoten, dessen Übergang zum aktuellen Knoten zu dieser höchsten Wahrscheinlichkeit geführt hat.

Dabei setzt sich der Wert der Funktion δ aus folgenden Faktoren zusammen:

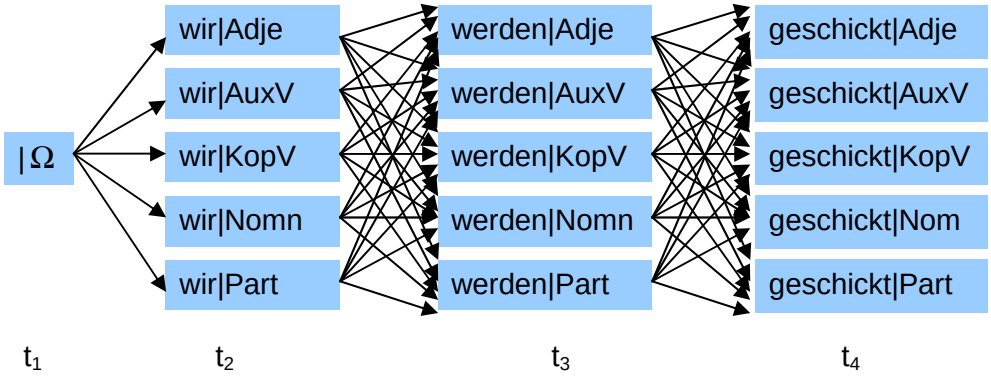
$\delta_i(t^k)$ Wahrscheinlichkeit des Pfades, der bis zum Vorgängerknoten reicht

P($w_{i+1}|t^j$) Emissionswahrscheinlichkeit des aktuellen Knotens

P($t^j|t^k$) Übergangswahrscheinlichkeit vom Vorgängerknoten zum aktuellen Knoten

Die Funktion ψ ermittelt die Kategorie, bei der die Funktion δ den Maximalwert liefert.

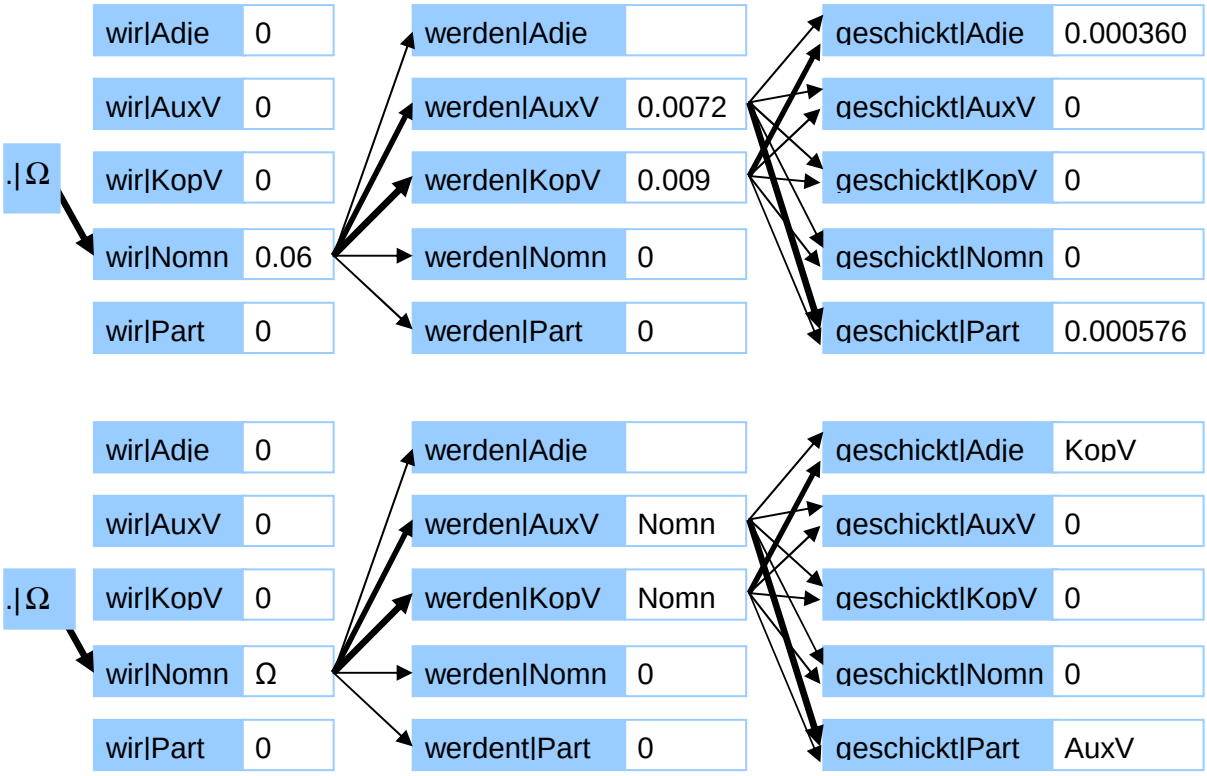
Es sind prinzipiell für jeden Knoten die Wahrscheinlichkeiten seines Erreichtwerdens von jedem Vorgängerknoten aus zu berechnen



Gespeichert wird aber bei jedem Knoten zum Zeitpunkt t nur der dabei gefundene maximale Werte, also

- die Wahrscheinlichkeit des wahrscheinlichsten Pfades, der zu diesem Knoten führt
- der Vorgängerknoten auf diesem Pfad.

Nur diese Werte werden zum nächsten Zeitpunkt t+1 weiterverwendet.



Die Ergebnisse der beiden Funktionen können in Arrays gespeichert werden. Allen (1995) verwendet für die Ergebnisse der Funktion δ ein Array namens SEQSCORE(j,i) und für die Ergebnisse der Funktion ψ ein Array namens BACKPTR(j,i). Das Array SEQSCORE speichert für jedes Wort (d.h. für jeden Zeitpunkt t) die Wahrscheinlichkeit, die der beste Pfad erreicht, der in jeder Kategorie endet. Das Array BACKPTR speichert zu jedem Knoten den besten Vorgängerknoten. Die Namen der Indizes (j und i) sind hier zum besseren Vergleich den Namen aus Manning und Schütze (2000) angeglichen.

		SEQSCORE(j,i) / $\delta_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.	0.0	0.0	0.0	0.0	0.0	1.0
2	wir	0.0	0.0	0.0	0.06	0.0	0.0
3	werden	0.0	0.0072	0.009	0.0	0.0	0.0
4	geschickt	0.000360	0.0	0.0	0.0	0.000576	0.0

		BACKPTR(j,i) / $\psi_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.						
2	wir	0	0	0	6	0	0
3	werden	0	4	4	0	0	0
4	geschickt	3	0	0	0	2	0

Das heißt z.B.: in der gegebenen Folge erreicht der Pfad, der über die Lesart *„geschickt“* als Adjektiv führt, als höchsten Wert die Wahrscheinlichkeit 0.000360, und zwar wenn der Vorgängerknoten der Knoten 3 (hier: KopV) ist. Der Pfad, der über die Lesart *„geschickt“* als Partizip führt, erreicht als höchsten Wert die Wahrscheinlichkeit 0.000576, und zwar wenn der Vorgänger der Knoten 2 (= AuxV) ist.

Im Termination-Schritt wird der beste Pfad an Hand von SEQSCORE festgestellt und dann an Hand von BACKPTR von hinten nach vorne verfolgt und dann ausgegeben. Im Beispiel hat am Ende der Knoten ‚geschickt|Part’ (SEQSCORE[4,5]) den höchsten Wert.
Die Pfadermittlung beginnt also in BACKPTR[4,5], wird von dort nach [3,2] verwiesen und von dort nach [2,4]. So ergibt sich der Pfad: <Nomn AuxV Part> als der beste Pfad für das Beispiel. Dieses Auslesen von hinten ist erforderlich, weil sich erst am Schluss zeigt, welche Lesart insgesamt die beste ist. So hat nach „*wir werden*“ die KopV-Lesart von werden zunächst den höchsten Wert. Diese Lesart ergibt aber in Kombination mit *„geschickt“* einen schlechteren Wert als die AuxV-Lesart.

		BACKPTR(j,i) / $\psi_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.						
2	wir	0	0	0	6	0	0
3	werden	0	4	4	0	0	0
4	geschickt	3	0	0	0	2	0

Im folgenden wird ein Tracing des Algorithmus vorgestellt. Dabei werden auch die Funktionen δ und ψ weiter erläutert.

7.1 Initialisierung

Der Initialisierungsschritt setzt die Werte für die Startkategorie bzw. für das Anfangszeichen, den Punkt. Nach der Initialisierung ergibt sich folgender Stand des Gitters:

.|Ω

.|Ω

Das entspricht folgendem Stand der Arrays SEQSCORE und BACKPTR:

		SEQSCORE(j,i) / $\delta_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.	0.0	0.0	0.0	0.0	0.0	1.0

		BACKPTR(j,i) / $\psi_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.						

In BACKPTR wird in diesem Schritt noch nichts eingetragen. So bleiben dort in der ersten Zeile die in der jeweiligen Implementierung gewählten Initialisierungswerte des Arrays stehen.

7.2 Induktion / Iteration

Der Induktions- bzw. Iterationsschritt ist folgendermaßen spezifiziert:

5 **comment:** Induction
6 **for** *i* := 1 **to** n **step** 1 **do**
7 **for** all tags t^j **do**
8 $\delta_{i+1}(t^j)$:= $\max_{1\le k\le T}$ [$\delta_i(t^k)$ x $P(w_{i+1}|t^j)$ x $P(t^j|t^k)$]
9 $\psi_{i+1}(t^j)$:= arg $\max_{1\le k\le T}$ [$\delta_i(t^k)$ x $P(w_{i+1}|t^j)$ x $P(t^j|t^k)$]
10 **end**
11 **end**

Die Schleifendurchläufe berechnen für jedes Wort die Werte der Funktionen $\delta_{i+1}(t^j)$ und $\psi_{i+1}(t^j)$. Dabei ermittelt $\delta_{i+1}(t^j)$ den maximalen Wert für jede Kategorie t^j und $\psi_{i+1}(t^j)$ die Kategorie, bei der der maximale Wert erreicht wurde.

Im folgenden Tracing sind die Werte für alle theoretisch konstruierbaren Pfade angegeben. Da im Viterbi-Algorithmus aber keine Werte gebraucht werden, die kleiner als das bereits gefundene Maximum sind, kann man die Berechnung der entsprechenden Pfade frühzeitig abbrechen. Die Abbruchsbedingungen sind implizit in der Spezifikation ‚max' enthalten. Bei der Programmierung sind sie entsprechend zu realisieren. In der Tracing-Tabelle sind alle Pfade, die nicht weiter berechnet werden müssen, durchgestrichen. Dies sind Pfade

- über Kanten, deren Wahrscheinlichkeit zu einem Gesamtwert unter dem bereits erreichten Maximum führt,
- über Knoten, deren Emissionswahrscheinlichkeit für das jeweils erreichte Wort = 0 ist (d.h. Knoten, die das entsprechende Wort nicht ausgeben) oder zu einem Gesamtwert unter dem bereits erreichten Maximum führt.

Im Feld ‚max' sind diese Pfade mit ‚-' gekennzeichnet.

Die in SEQSCORE zu speichernden Ergebnisse sind farbig markiert mit . Wenn sich der höchste Wert aus einem mit ‚-' markierten Pfad ergibt, bedeutet dies, dass in SEQSCORE der Initialisierungswert 0.0 stehen bleibt. In BACKPTR ist zu dem Maximalwert für ein Tag

die Nummer der Kategorie $\delta_i(t^k)$ (=Vorgängerkategorie) zu speichern, in deren Nachfolge sich dieser Wert ergibt (Markierung). Die Kategorien sind wie folgt durchnummeriert:

1 Adje,

2 AuxV,

3 KopV,

4 Nomn,

5 Part,

6 Ω

Falls der Maximalwert = 0 ist, wird in dem hier vorgestellten Beispiel in BACKPTR standardmäßig Kategorie 0 eingetragen (= keine gültige Vorgängerkategorie).

7.2.1 Wort 2: *wir*

Für das Wort 2 (*wir*) ergeben sich folgende Berechnungen:

δ_{i+1}	(t^j)	:= max	1≤k≤T	[$\delta_i(t^k)$		x P($w_{i+1}|t^j$)	x P($t^j|t^k$)]		
			Vorgängerknoten			aktueller Knoten			
				max. Prob.		Emission-Prob.	Transition-Probability		
δ_2	Adje	-	Adje	δ_1(Adje)	~~0.0~~	~~P(*wir*|Adje)~~	~~0.0~~	P(Adje|Adje)	0.2
wir		-	AuxV	δ_1(AuxV)	~~0.0~~		~~0.0~~	P(Adje|AuxV)	0.2
		-	KopV	δ_1(KopV)	~~0.0~~		~~0.0~~	P(Adje|KopV)	0.2
		-	Nomn	δ_1(Nomn)	~~0.0~~		~~0.0~~	P(Adje|Nomn)	0.05
		-	Part	δ_1(Part)	~~0.0~~		~~0.0~~	P(Adje|Part)	0.3
		-	Ω	δ_1(Ω)	1.0		~~0.0~~	P(Adje|Ω)	0.3
	AuxV	-	Adje	δ_1(Adje)	~~0.0~~	~~P(*wir*|Auxv)~~	~~0.0~~	P(AuxV|Adje)	0.1
		-	AuxV	δ_1(AuxV)	~~0.0~~		~~0.0~~	P(AuxV|AuxV)	0.3
		-	KopV	δ_1(KopV)	~~0.0~~		~~0.0~~	P(AuxV|KopV)	0.1
		-	Nomn	δ_1(Nomn)	~~0.0~~		~~0.0~~	P(AuxV|Nomn)	0.4
		-	Part	δ_1(Part)	~~0.0~~		~~0.0~~	P(AuxV|Part)	0.1
		-	Ω	δ_1(Ω)	1.0		~~0.0~~	P(AuxV|Ω)	0.2

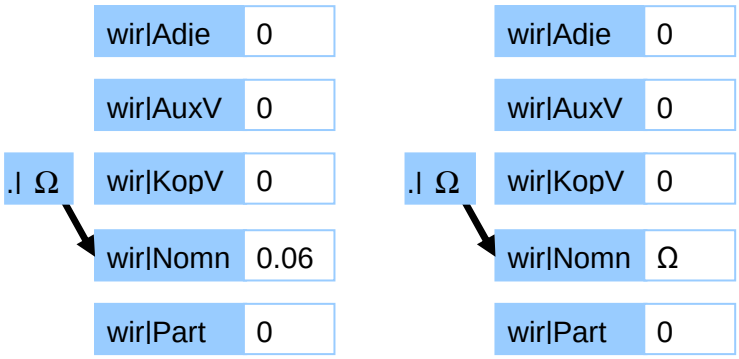
	KopV	-	Adje	δ_1 (Adje)	0.0	~~P(*wir*|KopV)~~	~~0.0~~	P(KopV|Adje)	0.1
		-	AuxV	δ_1 (AuxV)	0.0		~~0.0~~	P(KopV|AuxV)	0.1
		-	KopV	δ_1 (KopV)	0.0		~~0.0~~	P(KopV|KopV)	0.1
		-	Nomn	δ_1 (Nomn)	0.0		~~0.0~~	P(KopV|Nomn)	0.3
		-	Part	δ_1 (Part)	0.0		~~0.0~~	P(KopV|Part)	0.1
		-	Ω	$\delta_1(\Omega)$	1.0		~~0.0~~	P(KopV|Ω)	0.1
	Nomn	-	Adje	δ_1 (Adje)	~~0.0~~	P(*wir*|Nomn)	0.2	P(Nomn|Adje)	0.4
		-	AuxV	δ_1 (AuxV)	~~0.0~~		0.2	P(Nomn|AuxV)	0.1
		-	KopV	δ_1 (KopV)	~~0.0~~		0.2	P(Nomn|KopV)	0.4
		-	Nomn	δ_1 (Nomn)	~~0.0~~		0.2	P(Nomn|Nomn)	0.05
		-	Part	δ_1 (Part)	~~0.0~~		0.2	P(Nomn|Part)	0.1
		0.06	Ω	$\delta_1(\Omega)$	1.0		0.2	P(Nomn|Ω)	0.3
	Part	-	Adje	δ_1 (Adje)	~~0.0~~	~~P(*wir*|Part)~~	~~0.0~~	P(Part|Adje)	0.1
		-	AuxV	δ_1 (AuxV)	~~0.0~~		~~0.0~~	P(Part|AuxV)	0.2
		-	KopV	δ_1 (KopV)	~~0.0~~		~~0.0~~	P(Part|KopV)	0.1
		-	Nomn	δ_1 (Nomn)	~~0.0~~		~~0.0~~	P(Part|Nomn)	0.1
		-	Part	δ_1 (Part)	~~0.0~~		~~0.0~~	P(Part|Part)	0.3
		-	Ω	$\delta_1(\Omega)$	1.0		~~0.0~~	P(Part|Ω)	0.1
	Ω	-	Adje	δ_1 (Adje)	~~0.0~~	~~P(*wir*|P…)~~	~~0.0~~	P(Ω|Adje)	0.1
		-	AuxV	δ_1 (AuxV)	~~0.0~~		~~0.0~~	P(Ω|AuxV)	0.1
		-	KopV	δ_1 (KopV)	~~0.0~~		~~0.0~~	P(Ω|KopV)	0.1
		-	Nomn	δ_1 (Nomn)	~~0.0~~		~~0.0~~	P(Ω|Nomn)	0.1
		-	Part	δ_1 (Part)	~~0.0~~		~~0.0~~	P(Ω|Part)	0.1
		-	Ω	$\delta_1(\Omega)$	1.0		~~0.0~~	P(Ω|Ω)	0.1

Nach diesem Schritt haben die Arrays die folgenden Werte:

		SEQSCORE(j,i) / $\delta_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.	0.0	0.0	0.0	0.0	0.0	1.0
2	wir	0.0	0.0	0.0	0.06	0.0	0.0

		BACKPTR(j,i) / $\psi_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.						
2	wir	0	0	0	6	0	0

Dies entspricht folgendem Zustand des Graphen:



Wort 3: *werden*

Für das Wort 3 (*werden*) ergeben sich folgende Berechnungen:

δ_{i+1}	(t^j)	:= max	$1 \le k \le T$	$[\delta_i(t^k)$	x $P(w_{i+1}\|t^j)$		x $P(t^j\|t^k)$]	
			Vorgängerknoten		aktueller Knoten			
				max. Prob.		Emission-Prob.		Transition-Prob.	
δ_3	Adje	-	Adje	δ_2(Adje)	~~0.0~~	~~P(*werden*|Adje)~~	0.0	P(Adje|Adje)	0.2

w e r d e n		-	AuxV	δ_2(AuxV)	~~0.0~~		0.0	P(Adje|AuxV)	0.2
		-	KopV	δ_2(KopV)	~~0.0~~		0.0	P(Adje|KopV)	0.2
		-	Nomn	δ_2(Nomn)	0.06		0.0	P(Adje|Nomn)	0.05
		-	Part	δ_2(Part)	~~0.0~~		0.0	P(Adje|Part)	0.3
		-	Ω	$\delta_2(\Omega)$	~~0.0~~		0.0	P(Adje|Ω)	0.2
	AuxV	-	Adje	δ_2(Adje)	~~0.0~~	P(*werden*|Auxv)	0.3	P(AuxV|Adje)	0.1
		-	AuxV	δ_2(AuxV)	~~0.0~~		0.3	P(AuxV|AuxV)	0.3
		-	KopV	δ_2(KopV)	~~0.0~~		0.3	P(AuxV|KopV)	0.1
		0.0072	Nomn	δ_2(Nomn)	0.06		0.3	P(AuxV|Nomn)	0.4
		-	Part	δ_2(Part)	~~0.0~~		0.3	P(AuxV|Part)	0.1
		-	Ω	$\delta_2(\Omega)$	~~0.0~~		0.3	P(AuxV|Ω)	0.2
	KopV	-	Adje	δ_2(Adje)	~~0.0~~	P(*werden*|KopV)	0.5	P(KopV|Adje)	0.1
		-	AuxV	δ_2(AuxV)	~~0.0~~		0.5	P(KopV|AuxV)	0.1
		-	KopV	δ_2(KopV)	~~0.0~~		0.5	P(KopV|KopV)	0.1
		0.009	Nomn	δ_2(Nomn)	0.06		0.5	P(KopV|Nomn)	0.3
		-	Part	δ_2(Part)	~~0.0~~		0.5	P(KopV|Part)	0.1
		-	Ω	$\delta_2(\Omega)$	~~0.0~~		0.5	P(KopV|Ω)	0.1
	Nomn	-	Adje	δ_2(Adje)	~~0.0~~	~~P(*werden*|Nomn)~~	0.0	P(Nomn|Adje)	0.4
		-	AuxV	δ_2(AuxV)	~~0.0~~		0.0	P(Nomn|AuxV)	0.1
		-	KopV	δ_2(KopV)	~~0.0~~		0.0	P(Nomn|KopV)	0.4
		-	Nomn	δ_2(Nomn)	0.06		0.0	P(Nomn|Nomn)	0.05
		-	Part	δ_2(Part)	~~0.0~~		0.0	P(Nomn|Part)	0.1
		-	Ω	$\delta_2(\Omega)$	~~0.0~~		0.0	P(Nomn|Ω)	0.3
	Part	-	Adje	δ_2(Adje)	~~0.0~~	~~P(*werden*|Part)~~	0.0	P(Part|Adje)	0.1
		-	AuxV	δ_2(AuxV)	~~0.0~~		0.0	P(Part|AuxV)	0.2
		-	KopV	δ_2(KopV)	~~0.0~~		0.0	P(Part|KopV)	0.1
		-	Nomn	δ_2(Nomn)	0.06		0.0	P(Part|Nomn)	0.1
		-	Part	δ_2 (Part)	~~0.0~~		0.0	P(Part|Part)	0.3
		-	Ω	δ_2 (Ω)	~~0.0~~		0.0	P(Part|Ω)	0.1

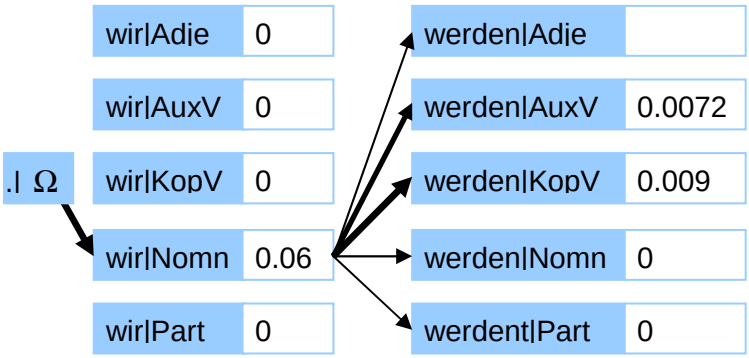
	Ω	-	Adje	δ_2 (Adje)	~~0.0~~	~~P(*werden*|P…)~~	0.0	P(Ω|Adje)	0.1
		-	AuxV	δ_2 (AuxV)	~~0.0~~		0.0	P(Ω|AuxV)	0.1
		-	KopV	δ_2(KopV)	~~0.0~~		0.0	P(Ω|KopV)	0.1
		-	Nomn	δ_2(Nomn)	0.06		0.0	P(Ω|Nomn)	0.1
		-	Part	δ_2(Part)	~~0.0~~		0.0	P(Ω|Part)	0.1
		-	Ω	$\delta_2(\Omega)$	~~0.0~~		0.0	P(Ω|Ω)	0.1

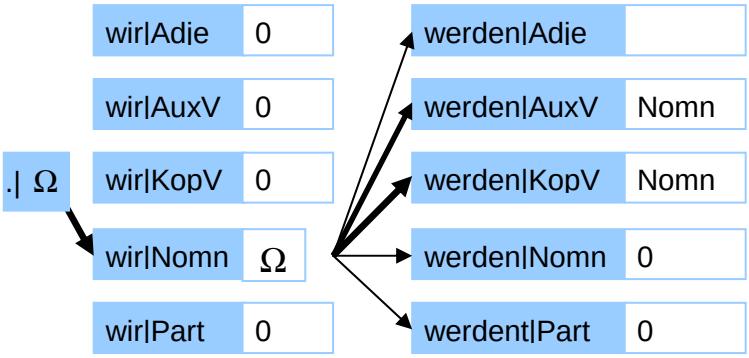
Nach diesem Schritt haben die Arrays die folgenden Werte:

		SEQSCORE(j,i) / $\delta_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.	0.0	0.0	0.0	0.0	0.0	1.0
2	wir	0.0	0.0	0.0	0.06	0.0	0.0
3	werden	0.0	0.0072	0.009	0.0	0.0	0.0

		BACKPTR(j,i) / $\psi_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.						
2	wir	0	0	0	6	0	0
3	werden	0	4	4	0	0	0

Dies entspricht folgendem Zustand des Graphen:





Wort 4: *geschickt*

Für das Wort 4 (*geschickt*) ergeben sich folgende Berechnungen:

δ_{i+1}	(t^j)	:= max	$1 \le k \le T$	$[\delta_i(t^k)$		x P(w_{i+1}|t^j)		x P(t^j|t^k)]	
			Vorgängerknoten			aktueller Knoten			
				max. Prob.		Emission-Prob.		Transition-Prob.	
δ_4	Adje	-	Adje	δ_3(Adje)	~~0.0~~	P(*geschickt* | Adje)	0.2	P(Adje|Adje)	0.2
g		0.000288	AuxV	δ_3(AuxV)	0.0072		0.2	P(Adje|AuxV)	0.2
e		0.000360	KopV	δ_3(KopV)	0.009		0.2	P(Adje|KopV)	0.2
s		-	Nomn	δ_3(Nomn)	~~0.0~~		0.2	P(Adje|Nomn)	0.05
c		-	Part	δ_3(Part)	~~0.0~~		0.2	P(Adje|Part)	0.3
h		-	Ω	$\delta_3(\Omega)$	~~0.0~~		0.2	P(Adje|Ω)	0.3
i	AuxV	-	Adje	δ_3(Adje)	~~0.0~~	~~P(~~ ~~*geschickt*~~	0.0	P(AuxV|Adje)	0.1
c		-	AuxV	δ_3(AuxV)	0.0072		0.0	P(AuxV|AuxV)	0.3
k									
t									

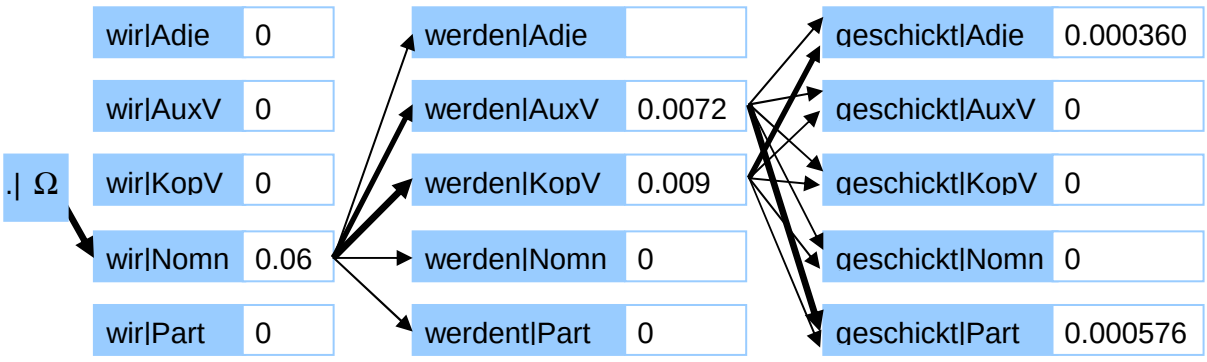
		-	KopV	δ_3 (KopV)	0.009	~~| Auxv)~~	0.0	P(AuxV|KopV)	0.1
		-	Nomn	δ_3 (Nomn)	~~0.0~~		0.0	P(AuxV|Nomn)	0.4
		-	Part	δ_3 (Part)	~~0.0~~		0.0	P(AuxV|Part)	0.1
		-	Ω	δ_3 (Ω)	~~0.0~~		0.0	P(AuxV|Ω)	0.2
	KopV	-	Adje	δ_3 (Adje)	~~0.0~~	~~P(*geschickt* | KopV)~~	0.0	P(KopV|Adje)	0.1
		-	AuxV	δ_3 (AuxV)	0.0072		0.0	P(KopV|AuxV)	0.1
		-	KopV	δ_3 (KopV)	0.009		0.0	P(KopV|KopV)	0.1
		-	Nomn	δ_3 (Nomn)	~~0.0~~		0.0	P(KopV|Nomn)	0.3
		-	Part	δ_3 (Part)	~~0.0~~		0.0	P(KopV|Part)	0.1
		-	Ω	δ_3 (Ω)	~~0.0~~		0.0	P(KopV|Ω)	0.1
	Nomn	-	Adje	δ_3 (Adje)	~~0.0~~	~~P(*geschickt* | Nomn)~~	0.0	P(Nomn|Adje)	0.4
		-	AuxV	δ_3 (AuxV)	0.0072		0.0	P(Nomn|AuxV)	0.1
		-	KopV	δ_3 (KopV)	0.009		0.0	P(Nomn|KopV)	0.4
		-	Nomn	δ_3 (Nomn)	~~0.0~~		0.0	P(Nomn|Nomn)	0.05
		-	Part	δ_3 (Part)	~~0.0~~		0.0	P(Nomn|Part)	0.1
		-	Ω	δ_3 (Ω)	~~0.0~~		0.0	P(Nomn|Ω)	0.3
	Part	-	Adje	δ_3 (Adje)	~~0.0~~	P(*geschickt* | Part)	0.4	P(Part|Adje)	0.1
		0.000576	AuxV	δ_3 (AuxV)	0.0072		0.4	P(Part|AuxV)	0.2
		0.000360	KopV	δ_3 (KopV)	0.009		0.4	P(Part|KopV)	0.1
		-	Nomn	δ_3 (Nomn)	~~0.0~~		0.4	P(Part|Nomn)	0.1
		-	Part	δ_3 (Part)	~~0.0~~		0.4	P(Part|Part)	0.3
		-	Ω	δ_3 (Ω)	~~0.0~~		0.4	P(Part)	0.1
	Ω	-	Adje	δ_3 (Adje)	~~0.0~~	~~P(*geschickt* | P…)~~	0.0	P(Ω|Adje)	0.1
		-	AuxV	δ_3 (AuxV)	0.0072		0.0	P(Ω|AuxV)	0.1
		-	KopV	δ_3 (KopV)	0.009		0.0	P(Ω|KopV)	0.1
		-	Nomn	δ_3 (Nomn)	0.0		0.0	P(Ω|Nomn)	0.1
		-	Part	δ_3 (Part)	~~0.0~~		0.0	P(Ω|Part)	0.1
		-	Ω	δ_3 (Ω)	~~0.0~~		0.0	P(Ω|Ω)	0.1

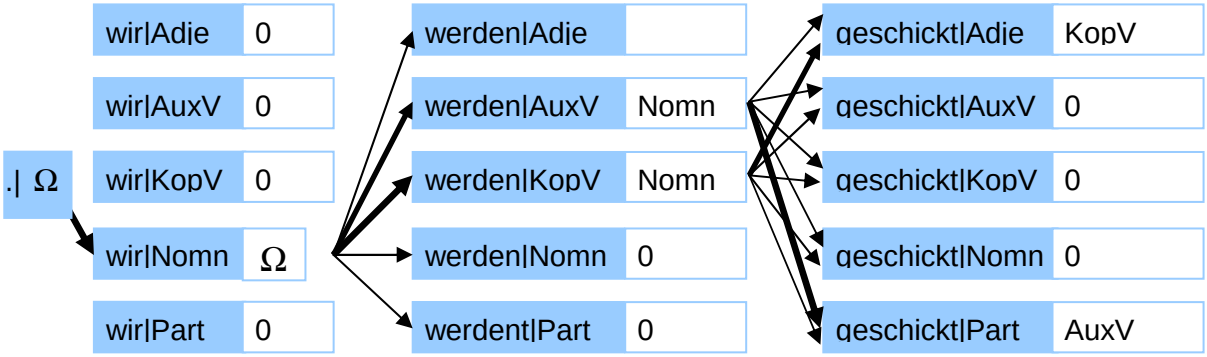
Nach diesem Schritt haben die Arrays die folgenden Werte:

		SEQSCORE(j,i) / $\delta_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.	0.0	0.0	0.0	0.0	0.0	1.0
2	wir	0.0	0.0	0.0	0.06	0.0	0.0
3	werden	0.0	0.0072	0.009	0.0	0.0	0.0
4	geschickt	0.000360	0.0	0.0	0.0	0.000576	0.0

		BACKPTR(j,i) / $\psi_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.						
2	wir	0	0	0	6	0	0
3	werden	0	4	4	0	0	0
4	geschickt	3	0	0	0	2	0

Dies entspricht folgendem Zustand des Graphen:





Terminierung und Pfadausgabe

Terminierung und Pfadausgabe sind wie folgt spezifiziert:

12 **comment:** Termination and path-readout

13 $X_{n+1} = \arg\max_{1 \le j \le T} \delta_{n+1}(j)$

14 **for** j := n **to** 1 **step** -1 **do**

15 $X_j = \psi_{j+1}(X_{j+1})$

16 **end**

17 $P(X_1, \ldots, X_n) = \max_{1 \le j \le T} \delta_{n+1}(t^j)$

$X_1, \ldots, X_n$ ist die Sequenz der Tags für die Wörter $w_1, \ldots, w_n$. Die Werte für das letzte Wort sind in den Arrays an Position n+1 gespeichert. Gemäß Zeile 13 soll für dieses Wort in SEQSCORE das Tag mit der höchsten Wahrscheinlichkeit identifiziert werden.

		SEQSCORE(j,i) / $\delta_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.	0.0	0.0	0.0	0.0	0.0	1.0
2	wir	0.0	0.0	0.0	0.06	0.0	0.0
3	werden	0.0	0.0072	0.009	0.0	0.0	0.0
4	geschickt	0.000360	0.0	0.0	0.0	0.000576	0.0

Dies ist das Tag „Part“.

Gemäß Zeile 14 bis 16 wird dann im Array BACKPTR Für dieses Tag rekursiv bis zum Satzanfang der beste Vorgänger gesucht. Dieser ist jeweils in BACKPTR notiert.

Gestartet wird in Zeile 13 bei X_4 = Part. Dann geht es in die Schleife:
Der beste Vorgänger X_3 von $BACKPTR_4$ (X_4=Part) = Kategorie 2 (= AuxV);
Der beste Vorgänger X_2 von $BACKPTR_3$ (X_3=AuxV) = Kategorie 4 (= Nomn);
Der beste Vorgänger X_1 von $BACKPTR_2$ (X_2=Nomn) = Kategorie 6 (= Ω).
So ergibt sich rückwärts die Sequenz der Part-of-Speech Tags:

		BACKPTR(j,i) / $\psi_j(t^j)$					
i		Adje	AuxV	KopV	Nomn	Part	Ω
		1	2	3	4	5	6
1	.						
2	wir	0	0	0	6	0	0
3	werden	0	4	4	0	0	0
4	geschickt	3	0	0	0	2	0

Schließlich wird in Zeile 17 noch die Wahrscheinlichkeit dieser Sequenz ausgegeben. Sie steht im Array SEQSCORE.

Literatur

Allen, James (1995): *Natural Language Understanding*. 2nd edition. Addison-Wesley Publishing Co.

Brants, Thorsten (1999). *Statistische Methoden in der Sprachverarbeitung*. Seminarskript 15. Juni 1999

Haenelt, Karin (2007). Der Viterbi-Algorithmus im Part-of-Speech Tagging. Kursfolien. 16.06.2007 ([1] 11.05.2002). http://kontext.fraunhofer.de/haenelt/kurs/folien/Haenelt_Viterbi-Algorithmus.pdf

Haenelt, Karin (2002): *Hidden Markov Models*. Kursfolien. 09.05.2002. 29 S. http://kontext.fraunhofer.de/haenelt/kurs/folien/HMM.htm oder .../HMM.ppt oder .../HMM-2.ps oder .../HMM-6.ps

Haenelt, Karin (2002): *Elementare Begriffe der Wahrscheinlichkeitstheorie für die Sprachverarbeitung.* Kursfolien. 04.05.2002. 32 S. http://kontext.fraunhofer.de/haenelt/kurs/folien/ElementarWahrscheinlichkeit.htm oder .../ElementarWahrscheinlichkeit.ppt oder .../ElementarWahrscheinlichkeit-2.ps oder .../ElementarWahrscheinlichkeit-6.ps

Manning, Christopher D.; Schütze, Hinrich (1999): *Foundations of Statistical Natural Language Processing.* Cambridge, Mass., London: The MIT Press. (vgl.: http://www.sultry.arts.usyd.edu.au/fsnlp)

Viterbi, Andrew J. (1967): Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In: *IEEE Transactions on Information Theory* IT-13, S. 1260-1269.

Version

19.10.2007

16.10.2007

16.06.2007

21.07.2004

11.05.2002