LI07: Hidden Markov Modelle und Part-of-Speech Tagging

Maciej Sumalvico
sumalvico@informatik.uni-leipzig.de

18. Mai 2017

Automatische Sprachverarbeitung

Wiederholung: Statistisches Sprachmodell

Ein statistisches Sprachmodell besteht allgemein aus:

- einer Funktion ϕ, die Wortsequenzen auf **Kontextklassen** abbildet,
- einer Wahrscheinlichkeitsverteilung von Wort gegeben Kontextklasse: $P(W|C)$.

Beispiel: **Bigramm-Modell** (Kontextklasse = letztes Wort)

"<s> Herr Schuhmann liest ein Buch. </s>"

$$P(\text{liest}|\phi(\text{<s>}, \text{Herr}, \text{Schuhmann})) = P(\text{liest}|\text{Schuhmann})$$

Automatische Sprachverarbeitung

Part-of-Speech-Tags als Kontextklassen

Beobachtung:

- manche Kombinationen immer noch sehr selten, z.B.: $P(\text{liest}|\text{Schuhmann})$,
- immer noch zu viel Information: es ist egal, welcher Name an der Stelle von "Schuhmann" steht.

Idee:

- ϕ gibt nur die **Wortart** des letzten Wortes zurück:

  "<s> Herr[N] Schuhmann[N] liest[V] ein[ART] Buch[N] .[$.] </s>"

$$P(\text{liest[V]}|\phi(\text{<s>}, \text{Herr[N]}, \text{Schuhmann[N]})) = P(\text{liest[V]}|\text{N})$$

Automatische Sprachverarbeitung

Part-of-Speech-Tags als Kontextklassen

Notation: Statt $P(\text{liest[V]}|\text{N})$, schreiben wir:

$$P(W_k = \text{“liest”}, T_k = \text{V}|\, T_{k-1} = \text{N})$$

Annahme: die Abhängigkeit vom Kontext betrifft nicht das Wort selbst, sondern nur seine grammatische Kategorie (Tag).

Dementsprechend:

$$\begin{aligned} P(W_k = \text{“liest”}, T_k = \text{V}|\, T_{k-1} = \text{N}) \quad &= \quad P(W_k = \text{“liest”}\,|\, T_k = \text{V}) \\ &\quad \cdot P(\,T_k = \text{V}|\, T_{k-1} = \text{N}) \end{aligned}$$

Automatische Sprachverarbeitung

Part-of-Speech-Tags als Kontextklassen

Es ergibt sich ein folgendes Sprachmodell:

$$P(W_1, \ldots, W_n; T_1, \ldots, T_n) = P(T_1)P(W_1|T_1) \prod_{i=2}^{n} P(T_i|T_{i-1})P(W_i|T_i)$$

Beobachtung:
Die Tagsequenz $\langle T_1, \ldots, T_n \rangle$ bildet eine **Markov-Kette**.

Problem: Die Tags sind im Text nicht explizit gegeben.

Aufgabe: Part-of-Speech Tagging

- colour[A] television is a great improvement
- blue is a nice colour[N]
- they colour[V] the walls red

Tag-Ambiguität im Brown-Corpus:

Anzahl Tags	Anzahl Wortformen
1	35340
2	4100
3	264
4	61
5	12
6	2
7	1

Automatische Sprachverarbeitung

Aufgabe: Part-of-Speech Tagging

- Der Ritter verteidigte[V] die Burg.
- Die verteidigte[A] Burg.

- Experten suchen nach Wegen[N] aus der Finanzkrise.
- Wegen[P] der Finanzkrise werden Investitionen gestoppt.

- Können[V] Sie mir helfen?
- Es geht ums Können[N], nicht ums Wollen.

Automatische Sprachverarbeitung

Hidden Markov Model

Ein **Hidden Markov Model** definiert eine WV auf zwei Sequenzen:

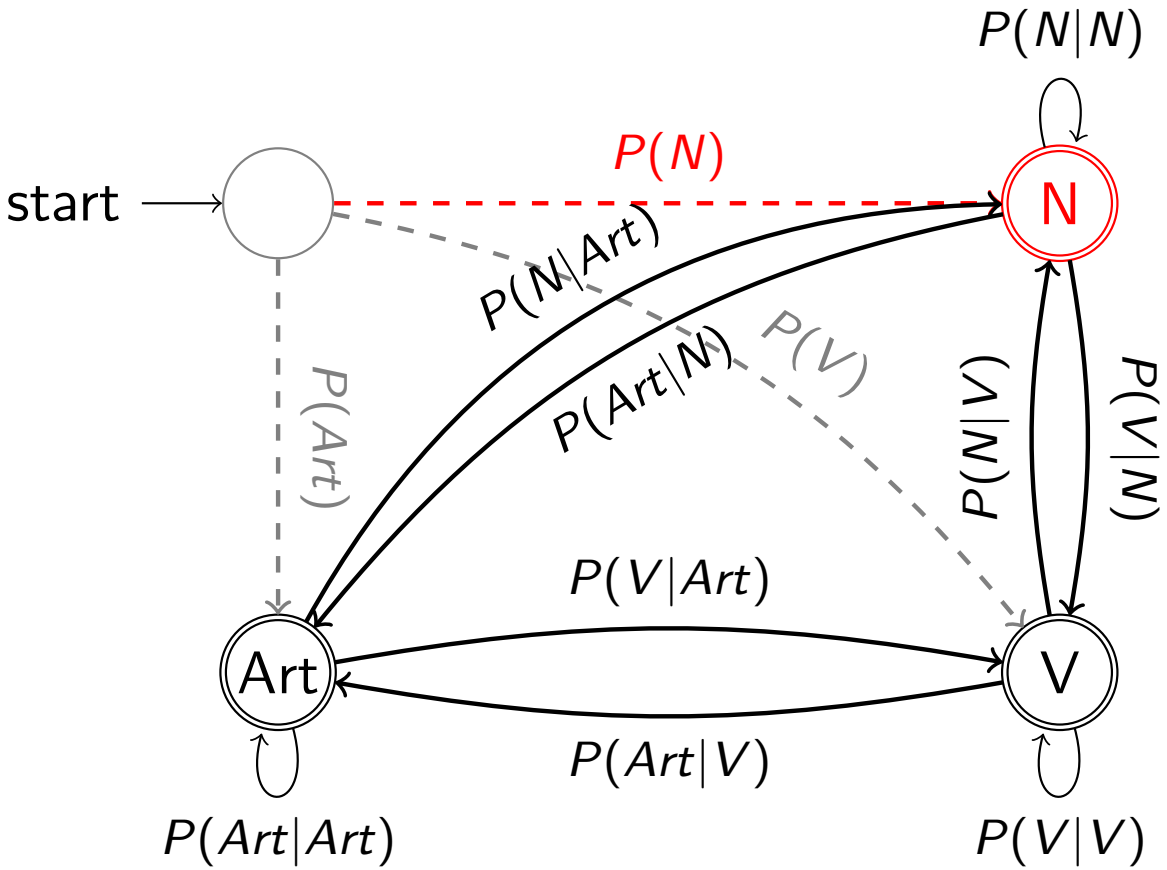
$$P(W_1, \ldots, W_n; T_1, \ldots, T_n) = P(T_1)P(W_1|T_1)\prod_{i=2}^{n} P(T_i|T_{i-1})P(W_i|T_i)$$

Begriffe:

- W_i – **Ausgabesymbole** (beobachtbar),
- T_i – **Versteckte Zustände** (nicht beobachtbar),
- $P(T_1)$ – **Anfangs**wahrscheinlichkeit,
- $P(T_i|T_{i-1})$ – **Übergangs**wahrscheinlichkeit,
- $P(W_i|T_i)$ – **Emissions**wahrscheinlichkeit.

Automatische Sprachverarbeitung

Beispiel



$P(\text{Herr}|N)$
$P(\text{Schuhmann}|N)$
$P(\text{Buch}|N)$

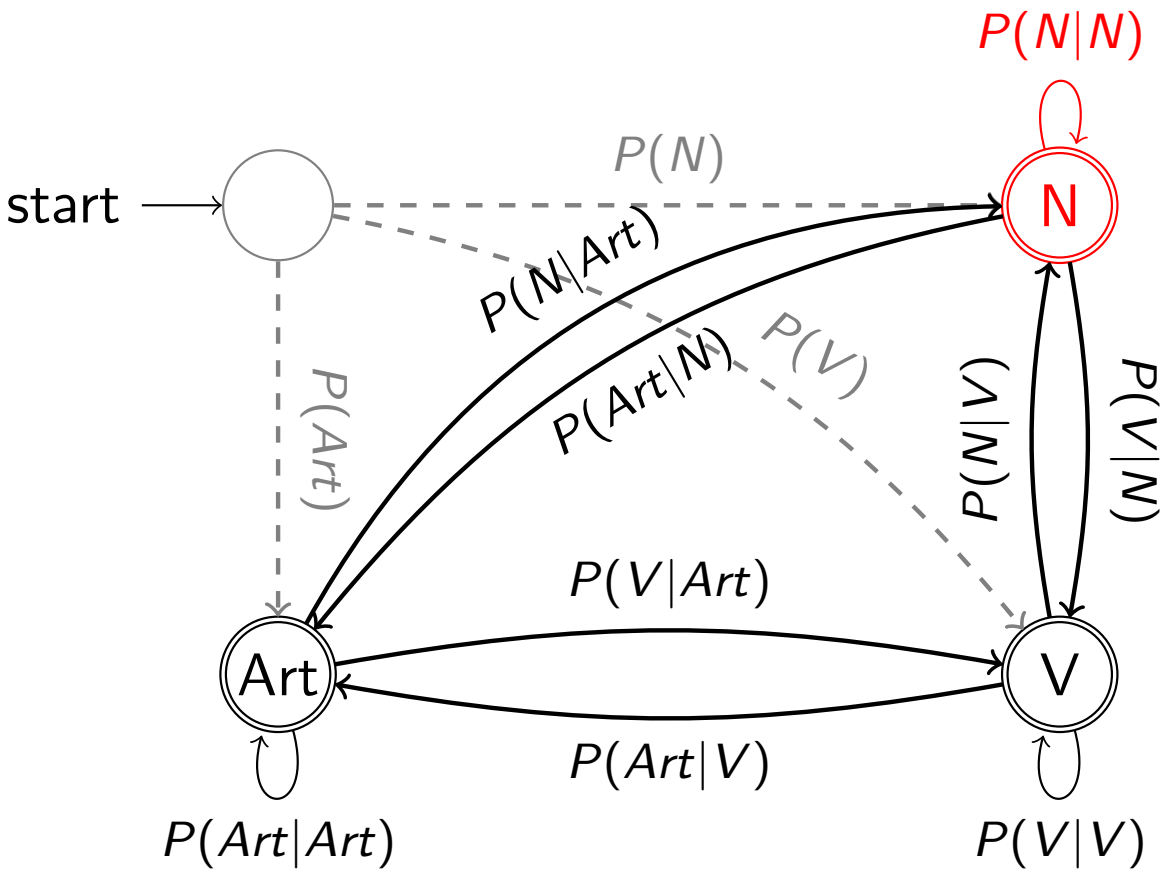
$P(\text{liest}|V)$

$P(\text{ein}|Art)$

Herr[N] Schuhmann[N] liest[V] ein[Art] Buch[N]

Automatische Sprachverarbeitung

Beispiel



$P(\text{Herr}|N)$
$P(\text{Schuhmann}|N)$
$P(\text{Buch}|N)$

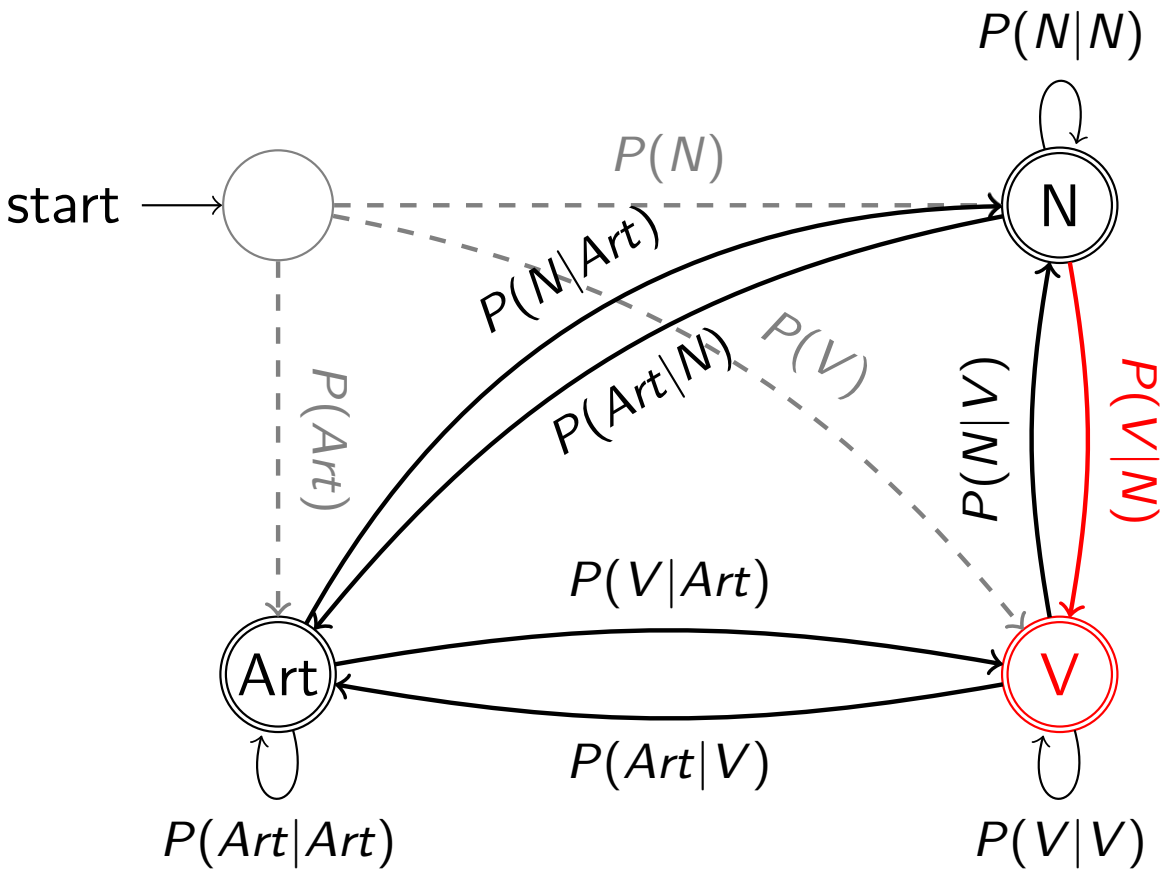
$P(\text{liest}|V)$

$P(\text{ein}|Art)$

Herr[N] Schuhmann[N] liest[V] ein[Art] Buch[N]

Automatische Sprachverarbeitung

Beispiel



$P(\text{Herr}|N)$
$P(\text{Schuhmann}|N)$
$P(\text{Buch}|N)$

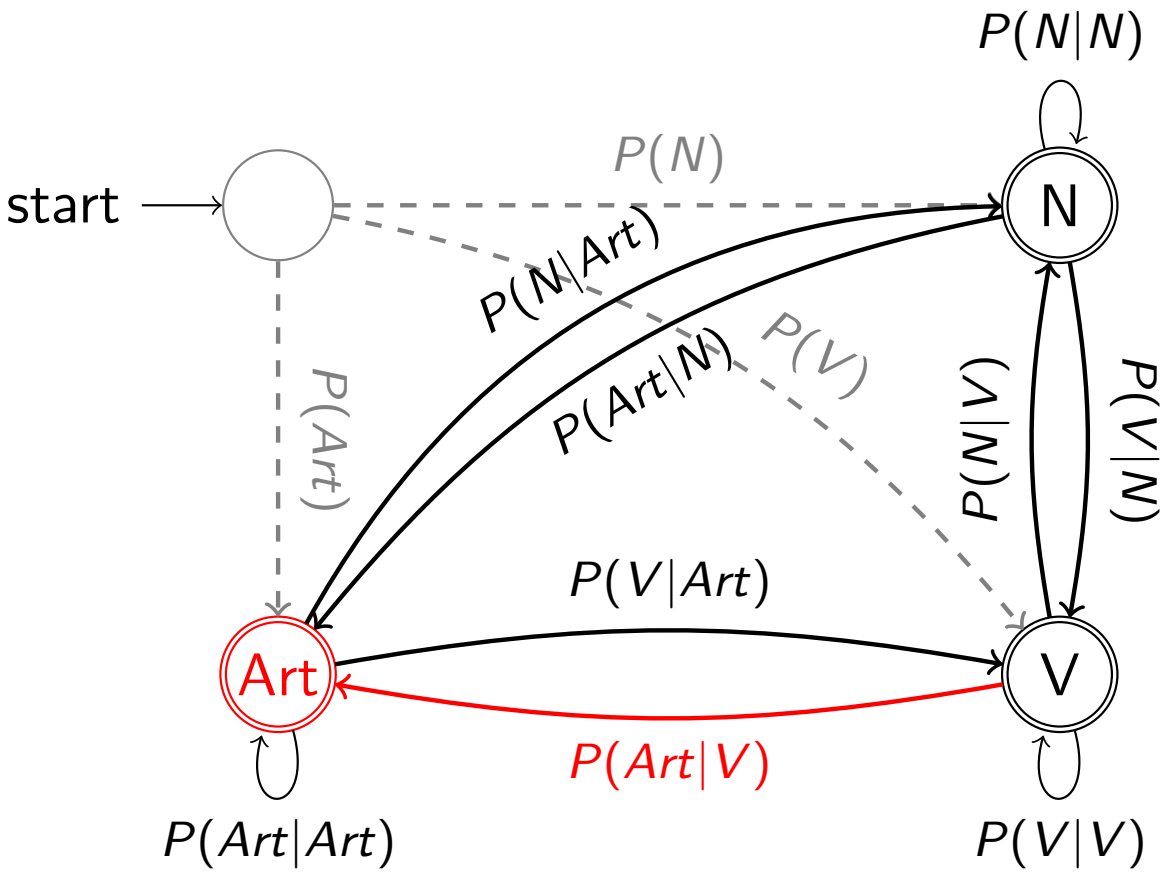
$P(\text{liest}|V)$

$P(\text{ein}|Art)$

Herr[N] Schuhmann[N] liest[V] ein[Art] Buch[N]

Automatische Sprachverarbeitung

Beispiel



$P(\text{Herr}|N)$
$P(\text{Schuhmann}|N)$
$P(\text{Buch}|N)$

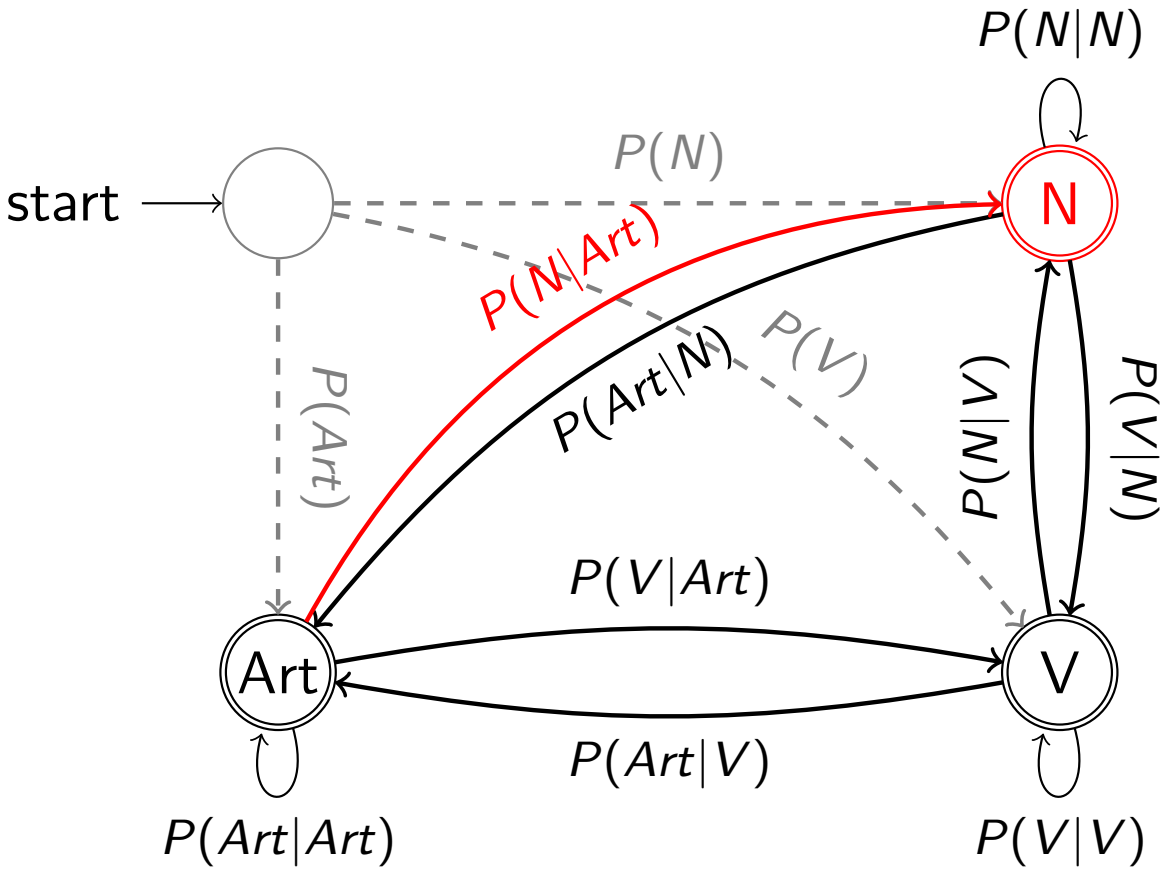
$P(\text{liest}|V)$

$P(\text{ein}|Art)$

Herr[N] Schuhmann[N] liest[V] ein[Art] Buch[N]

Automatische Sprachverarbeitung

Beispiel



$P(\text{Herr}|N)$
$P(\text{Schuhmann}|N)$
$P(\text{Buch}|N)$

$P(\text{liest}|V)$

$P(\text{ein}|Art)$

Herr[N] Schuhmann[N] liest[V] ein[Art] Buch[N]

Automatische Sprachverarbeitung

Die drei Standardaufgaben

1. Gegeben HMM und eine Folge von Ausgabesymbolen $\langle w_1, \ldots, w_n \rangle$, finde:

$$P(w_1, \ldots, w_n) = \sum_{t_1, \ldots, t_n} P(w_1, \ldots, w_n; t_1, \ldots, t_n)$$

$\rightarrow$ **Vorwärts/Rückwärts-Algorithmus**,

Automatische Sprachverarbeitung

Die drei Standardaufgaben

1. Gegeben HMM und eine Folge von Ausgabesymbolen $\langle w_1, \ldots, w_n \rangle$, finde:

$$P(w_1, \ldots, w_n) = \sum_{t_1, \ldots, t_n} P(w_1, \ldots, w_n; t_1, \ldots, t_n)$$

   $\rightarrow$ **Vorwärts/Rückwärts-Algorithmus**,

2. Gegeben HMM und eine Folge von Ausgabesymbolen $\langle w_1, \ldots, w_n \rangle$, finde:

$$\arg \max_{t_1, \ldots, t_n} P(w_1, \ldots, w_n; t_1, \ldots, t_n)$$

   $\rightarrow$ **Viterbi-Algorithmus**,

Die drei Standardaufgaben

1. Gegeben HMM und eine Folge von Ausgabesymbolen $\langle w_1, \ldots, w_n \rangle$, finde:

$$P(w_1, \ldots, w_n) = \sum_{t_1, \ldots, t_n} P(w_1, \ldots, w_n; t_1, \ldots, t_n)$$

   $\rightarrow$ **Vorwärts/Rückwärts-Algorithmus**,

2. Gegeben HMM und eine Folge von Ausgabesymbolen $\langle w_1, \ldots, w_n \rangle$, finde:

$$\arg \max_{t_1, \ldots, t_n} P(w_1, \ldots, w_n; t_1, \ldots, t_n)$$

   $\rightarrow$ **Viterbi-Algorithmus**,

3. Gegeben unannotiertes Korpus, finde die HMM-Parameter (Wahrscheinlichkeiten) $\rightarrow$ **Baum-Welch-Algorithmus**.

Vorwärts-Algorithmus

Ziel: Gegeben HMM und $\langle w_1, \ldots, w_n \rangle$, finde:

$$P(w_1, \ldots, w_n) = \sum_{t_1, \ldots, t_n} P(w_1, \ldots, w_n; t_1, \ldots, t_n)$$

Naiver Ansatz (alle Pfade berechnen) – Komplexität[1] $O(n|\mathcal{T}|^n)$

Idee: Nutze gemeinsame Fragmente von Pfaden.

[1] $\mathcal{T}$ – die Menge von möglichen PoS-Tags.

Vorwärts-Algorithmus

Sei $\alpha_{i,t}$ die Wahrscheinlichkeit, sich beim i-ten Ausgabesymbol im Zustand t zu befinden (*Vorwärts-Wahrscheinlichkeit*).

$$\alpha_{i,t} = \sum_{t_1,\ldots,t_{i-1} \in \mathcal{T}} P(w_1, \ldots, w_i; t_1, \ldots, t_{i-1}, t)$$

Beobachtungen:

$$\alpha_{1,t} = P(t)P(w_1|t) \quad (1)$$

$$\sum_{t \in \mathcal{T}} \alpha_{n,t} = P(w_1, \ldots, w_n) \quad (2)$$

$$\alpha_{i,t} = \sum_{t' \in \mathcal{T}} \alpha_{i-1,t'} P(t|t')P(w_i|t) \quad (3)$$

$\rightarrow$ Berechne $\alpha_{i,t}$ iterativ von $i = 1, \ldots, n$ für alle Zustände.

Komplexität: $O(n|\mathcal{T}|^2)$

Automatische Sprachverarbeitung

Rückwärts-Algorithmus

Sei $\beta_{i,t}$ die Wahrscheinlichkeit, beginnend am i-ten Symbol im Zustand t den Rest der Ausgabesequenz zu bekommen (analog wie α, aber von hinten).

$$\beta_{i,t} = \sum_{t_{i+1},\ldots,t_n \in \mathcal{T}} P(w_{i+1},\ldots,w_n; t_{i+1},\ldots,t_n | T_i = t)$$

Beobachtungen:

$$\beta_{n,t} = 1 \quad (4)$$

$$\sum_{t \in \mathcal{T}} P(t)P(w_1|t)\beta_{1,t} = P(w_1,\ldots,w_n) \quad (5)$$

$$\beta_{i,t} = \sum_{t' \in \mathcal{T}} P(t'|t)P(w_{i+1}|t')\beta_{i+1,t'} \quad (6)$$

Automatische Sprachverarbeitung

Viterbi-Algorithmus

Sei $\theta_{i,t}$ die Wahrscheinlichkeit des besten Pfades, der nach i Übergängen im Zustand t endet.

$$\theta_{i,t} = \max_{t_1,\ldots,t_{i-1} \in \mathcal{T}} P(w_1, \ldots, w_i; t_1, \ldots, t_{i-1}, t)$$

Beobachtungen:

$$\theta_{1,t} = P(t)P(w_1|t) \quad (7)$$

$$\theta_{i,t} = \max_{t' \in \mathcal{T}} \left[\theta_{i-1,t'} P(t|t')P(w_i|t)\right] \quad (8)$$

Zusätzlich sei:

$$\psi_{i,t} = \arg\max_{t' \in \mathcal{T}} \theta_{i-1,t'} P(t|t')P(w_i|t) \quad (9)$$

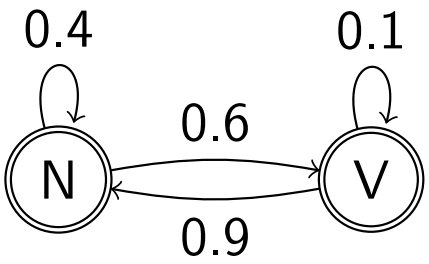
Der optimale Pfad ist aus θ's und ψ's rekonstruierbar:

$$t_n = \arg\max_{t \in \mathcal{T}} \theta_{n,t}$$

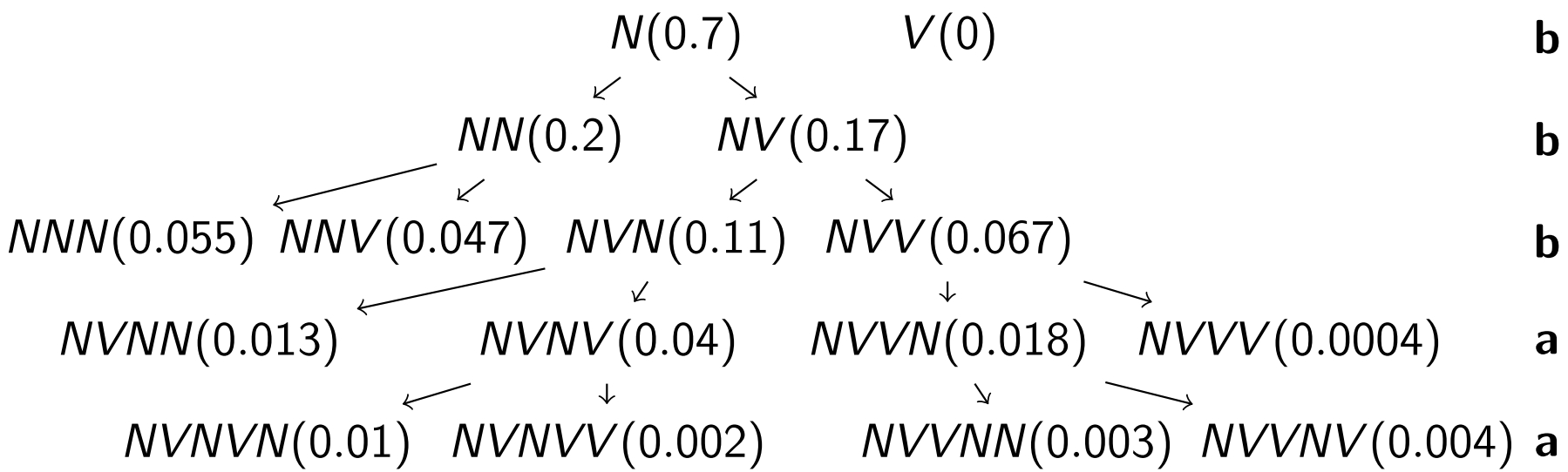
$$t_{i-1} = \psi_{i,t_i} \quad \text{für } 1 < i \leq n$$

Automatische Sprachverarbeitung

Beispiel



$$P(N) = 1 \qquad P(V) = 0$$
$$P(a|N) = 0.3 \qquad P(a|V) = 0.6$$
$$P(b|N) = 0.7 \qquad P(b|V) = 0.4$$



Hoffnungslose Verzweigungen werden nicht mehr verfolgt!

Automatische Sprachverarbeitung

Baum-Welch-Algorithmus

Gegeben:

- ein unannotierter Text: $\langle w_1, \ldots, w_n \rangle$,
- Menge $\mathcal{T}$ von **Zuständen**,
- Menge $A \subseteq \mathcal{T} \times \mathcal{T}$ von **möglichen Übergängen**,
- Menge $B \subseteq \mathcal{W} \times \mathcal{T}$ von **möglichen Emissionen**.

Gesucht:

- $P(T)$,
- $P(T|T')$,
- $P(W|T)$.

Baum-Welch-Algorithmus

Idee: "Expectation-Maximization"

1. Wähle zufällige Wahrscheinlichkeiten für mögliche Übergänge, Emissionen und Anfangszustände.
2. Wiederhole:
   - benutze das gegenwärtige HMM, um die Daten zu annotieren (Expectation),
   - berechne die optimalen Wahrscheinlichkeitswerte mittels ML-Estimation (Maximization).

Automatische Sprachverarbeitung

Baum-Welch-Algorithmus

Expectation:

Nicht nur die besten Werte, sondern *gewichteter Durchschnitt* über *alle* Werte!

$$
\begin{aligned}
Z &= P(w_1, \ldots, w_n) \\
c_{start}(t) &= \mathbb{E}_{T_1,\ldots,T_n|w_1,\ldots,w_n} \delta_{t,T_1} \\
c_{tr}(t', t) &= \mathbb{E}_{T_1,\ldots,T_n|w_1,\ldots,w_n} \sum_{i=1}^{n-1} \delta_{t',T_i} \delta_{t,T_{i+1}} \\
c_{em}(w, t) &= \mathbb{E}_{T_1,\ldots,T_n|w_1,\ldots,w_n} \sum_{i=1}^{n} \delta_{w,w_i} \delta_{t,T_i}
\end{aligned}
$$

mit: $\delta_{a,b} = \begin{cases} 1 & \text{falls } a = b \\ 0 & \text{falls } a \neq b \end{cases}$ (Kronecker-Delta)

Wenn das Korpus mehrere Sätze enthält, werden die c's über alle Sätze summiert!

Baum-Welch-Algorithmus

Expectation:

Nicht nur die besten Werte, sondern *gewichteter Durchschnitt* über *alle* Werte!

$$
\begin{aligned}
Z &= \sum_{t \in \mathcal{T}} \alpha_{n,t} \\
c_{start}(t) &= \mathbb{E}_{T_1,\ldots,T_n|w_1,\ldots,w_n} \delta_{t,T_1} \\
c_{tr}(t',t) &= \mathbb{E}_{T_1,\ldots,T_n|w_1,\ldots,w_n} \sum_{i=1}^{n-1} \delta_{t',T_i} \delta_{t,T_{i+1}} \\
c_{em}(w,t) &= \mathbb{E}_{T_1,\ldots,T_n|w_1,\ldots,w_n} \sum_{i=1}^{n} \delta_{w,w_i} \delta_{t,T_i}
\end{aligned}
$$

mit: $\delta_{a,b} = \begin{cases} 1 & \text{falls } a = b \\ 0 & \text{falls } a \neq b \end{cases}$ (Kronecker-Delta)

Wenn das Korpus mehrere Sätze enthält, werden die c's über alle Sätze summiert!

Baum-Welch-Algorithmus

Expectation:

Nicht nur die besten Werte, sondern *gewichteter Durchschnitt* über *alle* Werte!

$$
\begin{aligned}
Z &= \sum_{t \in \mathcal{T}} \alpha_{n,t} \\
c_{start}(t) &= \alpha_{1,t}\beta_{1,t}/Z \\
c_{tr}(t',t) &= \mathbb{E}_{T_1,\ldots,T_n|w_1,\ldots,w_n} \sum_{i=1}^{n-1} \delta_{t',T_i}\delta_{t,T_{i+1}} \\
c_{em}(w,t) &= \mathbb{E}_{T_1,\ldots,T_n|w_1,\ldots,w_n} \sum_{i=1}^{n} \delta_{w,w_i}\delta_{t,T_i}
\end{aligned}
$$

mit: $\delta_{a,b} = \begin{cases} 1 & \text{falls } a = b \\ 0 & \text{falls } a \neq b \end{cases}$ (Kronecker-Delta)

Wenn das Korpus mehrere Sätze enthält, werden die c's über alle Sätze summiert!

Baum-Welch-Algorithmus

Expectation:

Nicht nur die besten Werte, sondern *gewichteter Durchschnitt* über *alle* Werte!

$$
\begin{aligned}
Z &= \sum_{t \in \mathcal{T}} \alpha_{n,t} \\
c_{start}(t) &= \alpha_{1,t}\beta_{1,t}/Z \\
c_{tr}(t', t) &= \sum_{i=2}^{n} \alpha_{i-1,t'} P(t|t') P(w_i|t) \beta_{i,t}/Z \\
c_{em}(w, t) &= \mathbb{E}_{T_1,\ldots,T_n|w_1,\ldots,w_n} \sum_{i=1}^{n} \delta_{w,w_i}\delta_{t,T_i}
\end{aligned}
$$

mit: $\delta_{a,b} = \begin{cases} 1 & \text{falls } a = b \\ 0 & \text{falls } a \neq b \end{cases}$ (Kronecker-Delta)

Wenn das Korpus mehrere Sätze enthält, werden die c's über alle Sätze summiert!

Baum-Welch-Algorithmus

Expectation:

Nicht nur die besten Werte, sondern *gewichteter Durchschnitt* über *alle* Werte!

$$
\begin{aligned}
Z &= \sum_{t \in \mathcal{T}} \alpha_{n,t} \\
c_{start}(t) &= \alpha_{1,t}\beta_{1,t}/Z \\
c_{tr}(t', t) &= \sum_{i=2}^{n} \alpha_{i-1,t'} P(t|t') P(w_i|t) \beta_{i,t}/Z \\
c_{em}(w, t) &= \sum_{i=1}^{n} \alpha_{i,t} \delta_{w,w_i} \beta_{i,t}/Z
\end{aligned}
$$

mit: $\delta_{a,b} = \begin{cases} 1 & \text{falls } a = b \\ 0 & \text{falls } a \neq b \end{cases}$ (Kronecker-Delta)

Wenn das Korpus mehrere Sätze enthält, werden die c's über alle Sätze summiert!

Baum-Welch-Algorithmus

Maximization:

$$\hat{P}(t) = \frac{c_{start}(t)}{\sum_{t' \in \mathcal{T}} c_{start}(t')}$$

$$\hat{P}(t|t') = \frac{c_{tr}(t', t)}{\sum_{t'' \in \mathcal{T}} c_{tr}(t', t'')}$$

$$\hat{P}(w|t) = \frac{c_{em}(w, t)}{\sum_{w' \in \mathcal{W}} c_{em}(w', t)}$$

Automatische Sprachverarbeitung

Beispiel

Nach 0 Iterationen:

- colour[V] television[N] is[V] a[D] great[A] improvement[N]
- blue[A] is[V] a[D] nice[A] colour[N]
- they[PPR] colour[N] the[D] walls[N] red[N]
- we[PPR] have[V] a[D] blue[A] car[N]
- the[D] red[A] car[N] is[V] great[A]

	a	bl.	car	col.	gr.	ha.	im.	is	ni.	red	tel.	the	they	wa.	we
A		.36		.05	.34				.18	.06					
D	.54											.46			
N		.09	.24	.26			.14			.09	.13			.05	
PPR													.48		.52
V				.19		.62		.19							

	A	D	N	PPR	V
A	.17	.26	.26	.18	.17
D	.38	.21	.23	.08	.11
N	.28	.26	.19	.10	.17
PPR	.31	.28	.10	.28	.03
V	.09	.23	.28	.19	.22

	P(T)
A	.26
D	.06
N	.22
PPR	.24
V	.21

Automatische Sprachverarbeitung

Beispiel

Nach 1 Iterationen:

- colour[A] television[N] is[V] a[D] great[A] improvement[N]
- blue[A] is[V] a[D] nice[A] colour[N]
- they[PPR] colour[V] the[D] walls[N] red[N]
- we[PPR] have[V] a[D] blue[A] car[N]
- the[D] red[A] car[N] is[V] great[A]

	a	bl.	car	col.	gr.	ha.	im.	is	ni.	red	tel.	the	they	wa.	we
A		.27		.09	.31				.16	.17					
D	.60											.40			
N		.04	.26	.21			.13			.12	.13			.13	
PPR													.50		.50
V				.17		.21		.62							

	A	D	N	PPR	V
A	.02	.07	.69		.22
D	.70		.30		
N	.10	.12	.31		.47
PPR	.17		.27		.56
V	.22	.69	.10		

	P(T)
A	.19
D	.20
N	.12
PPR	.40
V	.09

Beispiel

Nach 10 Iterationen:

- colour[A] television[N] is[V] a[D] great[A] improvement[N]
- blue[A] is[V] a[D] nice[A] colour[N]
- they[PPR] colour[V] the[D] walls[N] red[A]
- we[PPR] have[V] a[D] blue[A] car[N]
- the[D] red[A] car[N] is[V] great[A]

	a	bl.	car	col.	gr.	ha.	im.	is	ni.	red	tel.	the	they	wa.	we
A		.25		.16	.25				.16	.25					
D	.60											.40			
N			.36	.10			.18				.18			.18	
PPR													.50		.50
V				.27		.18		.55							

	A	D	N	PPR	V
A			.75		.25
D	.80		.20		
N	.33				.67
PPR					1
V	.20	.80			

	P(T)
A	.40
D	.20
N	
PPR	.40
V	

Warum funktioniert es?

Mehrdeutigkeit nicht ganz so schlimm:

- viele Wörter sind eindeutig,
- auch mehrdeutige Wörter können nicht *alle* Tags annehmen.

$\rightarrow$ "sinnvolle" Tagfolgen dominieren die Übergangszählungen c_{tr},
$\rightarrow$ plausible Übergangswahrscheinlichkeiten,
$\rightarrow$ korrekte Auflösung von Mehrdeutigkeiten aus dem Kontext.

Automatische Sprachverarbeitung

Sprachmodell-Interpolation

$$
\begin{aligned}
P(W_k|W_{k-1}, W_{k-2}) &= \lambda_1 P_1(W_k) \\
&\quad +\lambda_2 P_2(W_k|W_{k-1}) \\
&\quad +\lambda_3 P_3(W_k|W_{k-1}, W_{k-2})
\end{aligned}
$$

Aufgabe: die λ-Koeffizienten automatisch bestimmen.

Automatische Sprachverarbeitung

Sprachmodell-Interpolation

Betrachten wir ein HMM mit Zuständen der Form $\langle k, w, w' \rangle$:

- $k \in \{0, 1, 2, 3\}$,
- $w, w' \in \mathcal{W}$.

Emissionswahrscheinlichkeiten:[2]

$$P(w|\langle 0, w, w' \rangle) = 1$$
$$P(\epsilon|\langle i, w, w' \rangle) = 1 \qquad i \in \{1, 2, 3\}$$

Übergangswahrscheinlichkeiten:

$$P(\langle 1, w, w' \rangle|\langle 0, w, w' \rangle) = \lambda_1 \qquad P(\langle 0, w'', w \rangle|\langle 1, w, w' \rangle) = P_1(w'')$$
$$P(\langle 2, w, w' \rangle|\langle 0, w, w' \rangle) = \lambda_2 \qquad P(\langle 0, w'', w \rangle|\langle 2, w, w' \rangle) = P_2(w''|w)$$
$$P(\langle 3, w, w' \rangle|\langle 0, w, w' \rangle) = \lambda_3 \qquad P(\langle 0, w'', w \rangle|\langle 3, w, w' \rangle) = P_3(w''|w, w')$$

[2] ϵ – leeres Symbol (oder – wenn nicht erwünscht – ein "Dummy"-Symbol, das vor jedem Zeichen der Ausgabesequenz eingefügt wird).

Sprachmodell-Interpolation

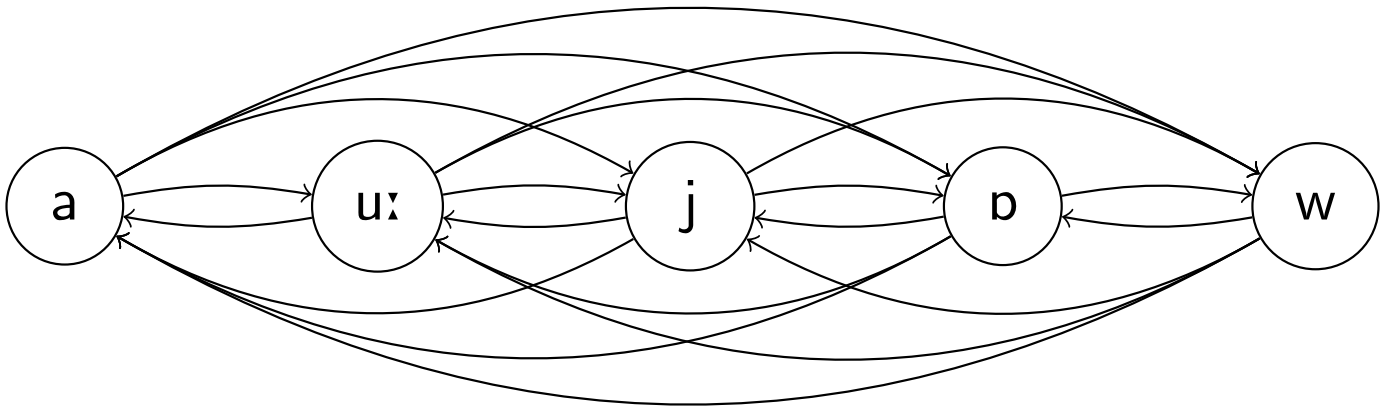
Anmerkungen:

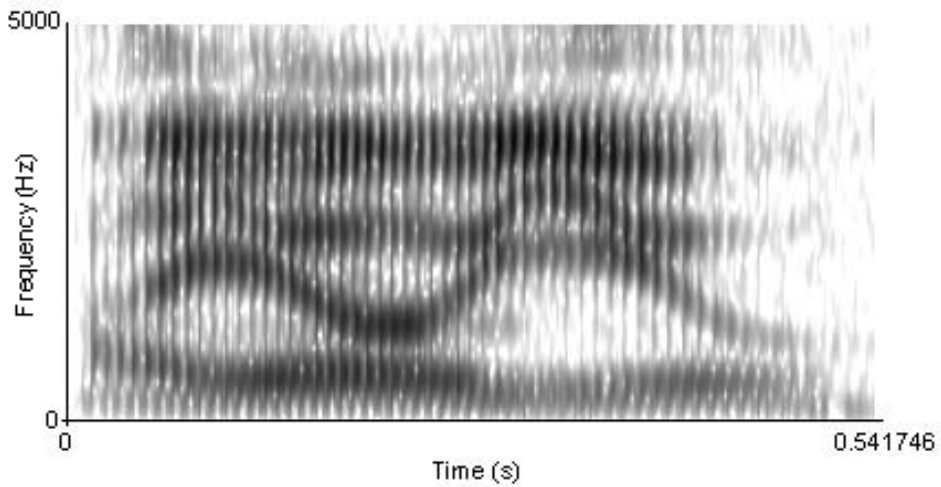
- die Sprachmodell-Wahrscheinlichkeiten P_1, P_2, P_3 und die Emissionswahrscheinlichkeiten sind fixiert,
- die Übergangswahrscheinlichkeiten λ_i werden von vielen Übergängen geteilt (*parameter tying*),
- Zählungen für die λ_i's im *Expectation*-Schritt werden über die verschiedenen Übergänge summiert.

Automatische Sprachverarbeitung

HMM-Anwendungen

Spracherkennung

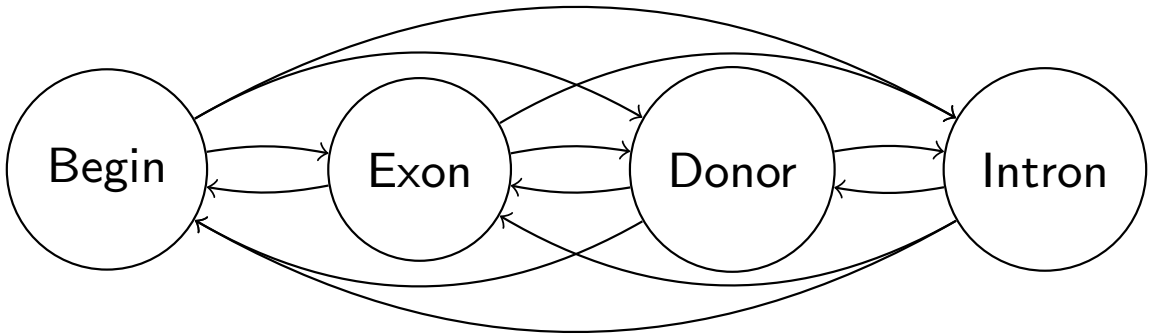




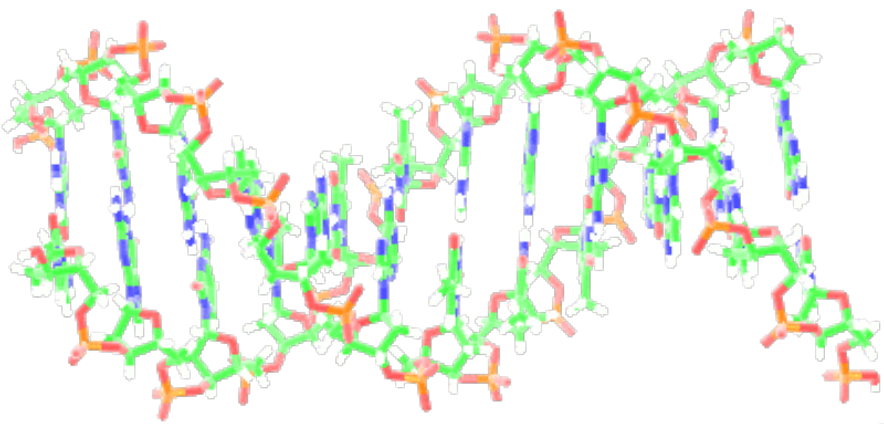
Automatische Sprachverarbeitung

HMM-Anwendungen

Generkennung in DNA-Sequenzen

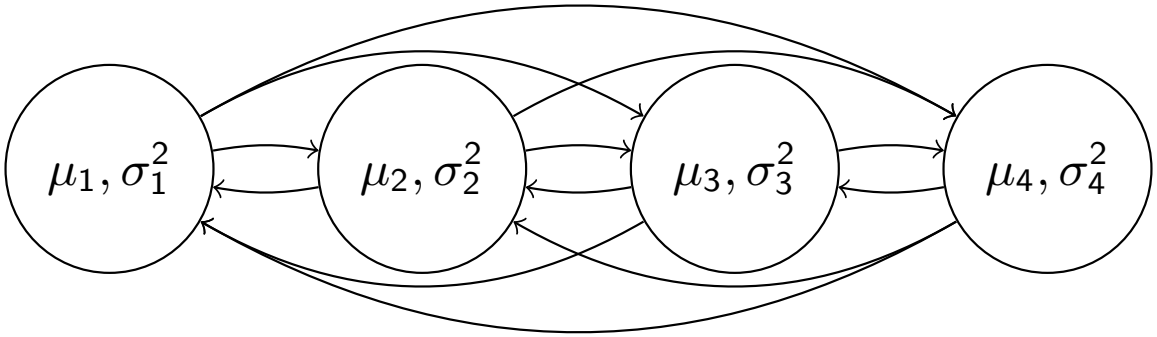


...AAAGCATGCATTTAACGAGTGCATCAGGACTCCATACGTAATGCCG...

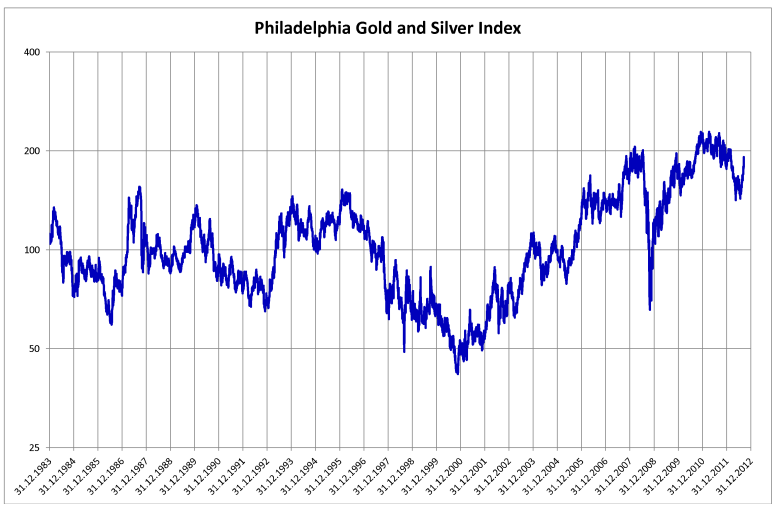


HMM-Anwendungen

Vorhersage von Aktienkursen



(μ_i, σ_i^2) – Parameter der Gauss-Verteilung.



Ergänzende Literatur

C. Manning und H. Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press: Cambridge (Mass.), 1999 (^{3}2000).

D. Jurafsky und J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*, Prentice Hall: San Francisco, 2000.

F. Jelinek, *Statistical Methods for Speech Recognition* (Kap. 2), MIT Press: Cambridge (Mass.), 1997 (^{3}2001).

Automatische Sprachverarbeitung