EM-Training für Wortart-Tagging

1. Wozu benutzen wir EM-Training?

2. Ist EM-Training überwachtes Lernen oder unüberwachtes Lernen?

3. Welche Resourcen braucht ein EM-Training?

4. Erkläre die Schritte für EM-Training.

5: Was sind die Parametern, die wir dabei schätzen wollen?

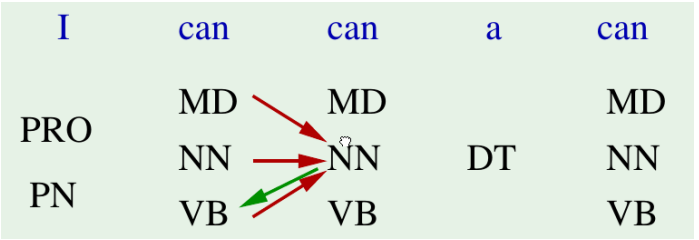
text: I love dog. Lexicon von Wortart
PRO, V/NN, NN

p(ti| t_i-k,...t_i-1) , p(w_i, t_i)

EM-Training

Lösung: Expectation-Maximization-Training

(maximiert iterativ die Wahrscheinlichkeit der Trainingsdaten)



1. gegeben
   - ein nicht annotiertes Trainingskorpus
   - ein Lexikon mit möglichen Wortarten von Wörtern
2. Initialisiere das HMM uniform (abgesehen davon, dass $p(w|t) = 0$ falls w im Lexikon enthalten und t keines seiner möglichen Tags ist)
3. Benutze das HMM, um das Trainingskorpus zu annotieren und extrahiere die Taghäufigkeiten (E-Schritt)
4. Schätze die HMM-Parameter aus den Taghäufigkeiten neu (M-Schritt)
5. weiter mit E-Schritt (bis irgendein Abbruchkriterium erfüllt ist)

$$p(w|t) = \frac{f(t,w)}{\sum_{w'} f(t,w')} \qquad p(t|t_1,...,t_k) = \frac{f(t_1,...,t_k,t)}{\sum_{t'} f(t_1,...,t_k,t')}$$

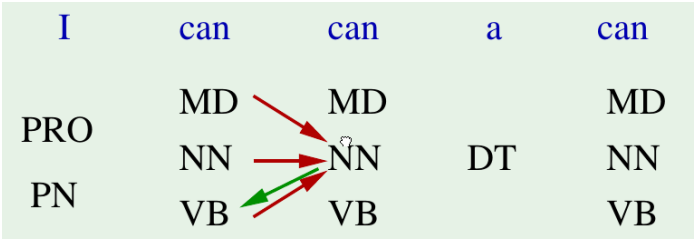
I	can	can	a	can
PRO	MD	VB	DT	NN

$$p(w|t) = \frac{f(t,w)}{\sum_{w'} f(t,w')} \qquad p(t|t_1,...,t_k) = \frac{f(t_1,...,t_k,t)}{\sum_{t'} f(t_1,...,t_k,t')}$$

f(tag,wort)
f(PRO, I) : 40
f(CAN, MD) : 16
...

f(tags, tag)
f(PRO, <s>) : 33
f(MD, MD) : 0
...

EM-Training



Variante 1: Wir benutzen den Viterbi-Algorithmus zum Taggen.

⇒ Nur die wahrscheinlichste Tagfolge wird berücksichtigt.
Alle anderen Tagfolgen werden ignoriert.

⇒ Am Anfang des Trainings gibt es aber noch keine eindeutige beste Tagfolge.

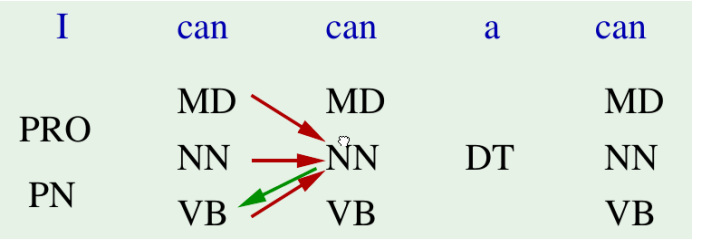
⇒ Das Training funktioniert deshalb so nicht.

Tag-Ergebnisse:
PRO MD MD DT MD , p(T|W) = ...
PRO NN MD DT MD , p(T|W) = ...
PRO ...
PN ....

Lösung

- Alle Tagfolgen bei der Extraktion der Taghäufigkeiten berücksichtigen.
- Jede Tagfolge wird dabei mit ihrer Wahrscheinlichkeit gewichtet, so dass doppelt so wahrscheinliche Tagfolgen doppelt so viel zu den extrahierten Häufigkeiten beitragen.

Gewichtung der Tagfolgen



Die Tagfolgen sollen so **gewichtet** werden, dass die Summe der Gewichte 1 ergibt (⇒ Wahrscheinlichkeitsverteilung)

Wir berechnen daher die Aposteriori-Wahrscheinlichkeit jeder Tagfolge T (= bedingte Wahrscheinlichkeit von T gegeben die Wortfolge W)

$$p(T|W) = \frac{p(T,W)}{p(W)} = \frac{p(T,W)}{\sum_{T'} p(T',W)}$$

Aus jeder Tagfolge werden die Tag-Tag-Häufigkeiten und die Tag-Wort-Häufigkeiten extrahiert, mit dem Gewicht der Tagfolge (Aposteriori-Wahrscheinlichkeit) multipliziert und aufsummiert.

Mit den so erhaltenen **erwarteten Häufigkeiten** werden die HMM-Parameter neu geschätzt.

Da die Aufzählung aller möglichen Tagfolgen zu ineffizient ist, benutzen wir wie beim Viterbi-Algorithmus *dynamische Programmierung*

⇒ Forward-Backward-Algorithmus

Tag-Ergebnisse:
PRO MD MD DT MD , p(T|W) = ...
PRO NN MD DT MD , p(T|W) = ...
PRO ...
PN ....

Tag-Tag-HF
erwartete_f(PRO, MD) = f(PRO, MD) * p(T1|W1) +
f(PRO, MD) * p(T2|W2) + ...

Tag-Wort-HF
erwartet_f(PRO,I) = f(PRO,I) * p(T1|W1) +
f(PRO,I) * p(T2|W2) + ...

Neuschätzung der HMM-Parameter

Hier müssen wir den Korpus nicht taggen, dann die Häuftigkeiten neu zählen, dann die Parametern neu schätzen, sondern benutzen wir die Formel mit Forward-Backward-Algorithmus um die Parameter neu zu schätzen.

$$p(w|t) = \frac{f_{tw}}{\sum_{w'} f_{tw'}}$$

$$p(t'|t) = \frac{f_{tt'}}{\sum_{t''} f_{tt''}}$$

$$\gamma_t(k) = \frac{\alpha_t(k)\,\beta_t(k)}{\alpha_{\langle s\rangle}(n+1)}$$

Aufsummierung der Aposteriori-Wahrscheinlichkeiten über alle Sätze **w** im Korpus C und alle Wortpositionen k zu erwarteten Häufigkeiten:

$$p(w|t) = \frac{f(t,w)}{\sum_{w'} f(t,w')} \qquad f_{tw} = \sum_{\mathbf{w}\in C} \sum_{1\le k\le|\mathbf{w}|:w_k=w} \gamma_t(k,\mathbf{w})$$

Die (Aposteriori-)Wahrscheinlichkeit $P(t_k = t|w_1^n) = \gamma_t(k)$

$$p(t|t_1,\ldots,t_k) = \frac{f(t_1,\ldots,t_k,t)}{\sum_{t'} f(t_1,\ldots,t_k,t')} \qquad f_{tt'} = \sum_{\mathbf{w}\in C} \sum_{k=1}^{n} \gamma_{tt'}(k,\mathbf{w})$$

$P(t_k = t, t_{k+1} = t'|w_1^n) = \gamma_{tt'}(k)$

Der Ausdruck $\sum_{1\le k\le n:w_k=w} \gamma_t(k)$ summiert über alle Positionen $k \in \{1,2,\ldots,n\}$ mit $w_k = w$. Man kann unter Verwendung der Indikatorfunktion auch schreiben: $\sum_{1\le k\le n} \gamma_t(k)\mathbb{1}_{w_k=w}$.

$\gamma_{tt'}(k,\mathbf{w})$ ist der Wert von $\gamma_{tt'}(k)$ für den Satz **w**.

$$= \frac{\alpha_t(k)\; p(t'|t)\; p(w_{k+1}|t')\; \beta_{t'}(k+1)}{\alpha_{\langle s\rangle}(n+1)}$$

Aufsummierung der Aposteriori-Wahrscheinlichkeiten über alle Sätze **w** im Korpus C und alle Wortpositionen k zu erwarteten Häufigkeiten:

$$f_{tw} = \sum_{\mathbf{w} \in C} \sum_{1 \leq k \leq |\mathbf{w}| : w_k = w} \gamma_t(k, \mathbf{w})$$

$$f_{tt'} = \sum_{\mathbf{w} \in C} \sum_{k=1}^{n} \gamma_{tt'}(k, \mathbf{w})$$

Der Ausdruck $\sum_{1 \leq k \leq n : w_k = w} \gamma_t(k)$ summiert über alle Positionen $k \in \{1, 2, \ldots, n\}$ mit $w_k = w$. Man kann unter Verwendung der Indikatorfunktion auch schreiben: $\sum_{1 \leq k \leq n} \gamma_t(k) \mathbb{1}_{w_k = w}$.

$\gamma_{tt'}(k, \mathbf{w})$ ist der Wert von $\gamma_{tt'}(k)$ für den Satz **w**.

f(tag,wort)
f(PRO, I) : 40
f(CAN, MD) : 16
...

f(tags, tag)
f(PRO, <s>) : 33
f(MD, MD) : 0
...

Sentence1: the dog is hungry
DT NN VB ADJ
DT VB VB ADJ
....
Sentence2: dog loved fish
NN VBD NN
NN VBN VB
....
Sentence3: cat does not want to eat
NN MD RB VB P MD
.....

To compute f(NN, DOG): look at each sentence w, look at each position k in the sentence, if there is "DOG" at position k, then compute **y_NN(k)** of w, sum them up.

To compute f(DT,NN): look at each sentence w, for each position k in the sentence, compute every **y_DT,NN(k)** and sum them up.

How to compute y_t(k) and y_t,t'(k) of a sentence w ?
> aus Forward und Backward-WK

4. In EM-Training, wie werden die Parametern initializiert?
#uniform
5. Was wird in E-Schritt gemacht?
6. Was wird in M-Schritt gemacht?
7. Benutzen wir Viterbi um den Korpus zu annotieren?

8.Wofür brauchen wir Forward und Backward-WK in EM-Training?

9. Wie schätzen wir die Häufigkeiten neu mit Hilfe von Forward-Backward-Algorithmus? (Formel und erkläre was die Formel bedeutet)

$$\gamma_t(k) = \frac{\alpha_t(k)\,\beta_t(k)}{\alpha_{\langle s\rangle}(n+1)}$$

$$\gamma_{tt'}(k) = \frac{\alpha_t(k)\,p(t'|t)\,p(w_{k+1}|t')\,\beta_{t'}(k+1)}{\alpha_{\langle s\rangle}(n+1)}$$

Formeln für die rekursive Berechnung im Fall des Bigramm-Taggers:

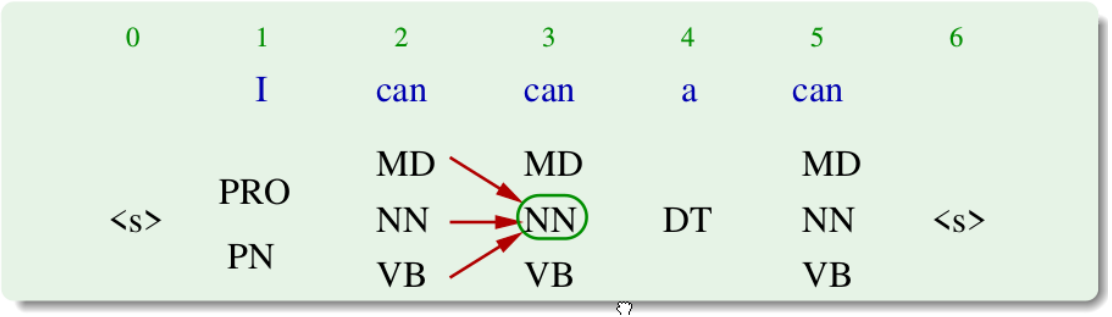
Formel merken

$$\alpha_t(0) = \begin{cases} 1 & \text{falls } t = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$$

$$\alpha_t(k) = \sum_{t' \in T} \alpha_{t'}(k-1)\, p(t|t')\, p(w_k|t) \quad \text{für } 0 < k \leq n+1$$

Bis auf das Summenzeichen ist der Forward-Algorithmus identisch zum Viterbi-Algorithmus.

Beispiel: Forward-Algorithmus



Berechnung der Forward-Wahrscheinlichkeit des Tags NN an Position 3:

$$\begin{aligned} \alpha_{NN}(3) = {} & \alpha_{MD}(2)\, p(NN|MD)\, p(can|NN) + \\ & \alpha_{NN}(2)\, p(NN|NN)\, p(can|NN) + \\ & \alpha_{VB}(2)\, p(NN|VB)\, p(can|NN) \end{aligned}$$

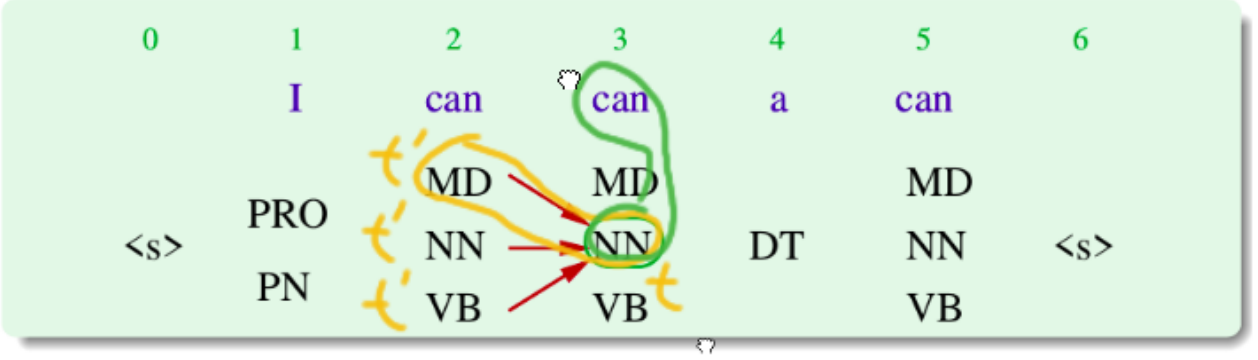
Formeln für die rekursive Berechnung im Fall des Bigramm-Taggers:

$$\alpha_t(0) = \begin{cases} 1 & \text{falls } t = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$$

$$\alpha_t(k) = \sum_{t' \in T} \alpha_{t'}(k-1)\, p(t|t')\, p(w_k|t) \quad \text{für } 0 < k \leq n+1$$

Bis auf das Summenzeichen ist der Forward-Algorithmus identisch zum Viterbi-Algorithmus.

Beispiel: Forward-Algorithmus



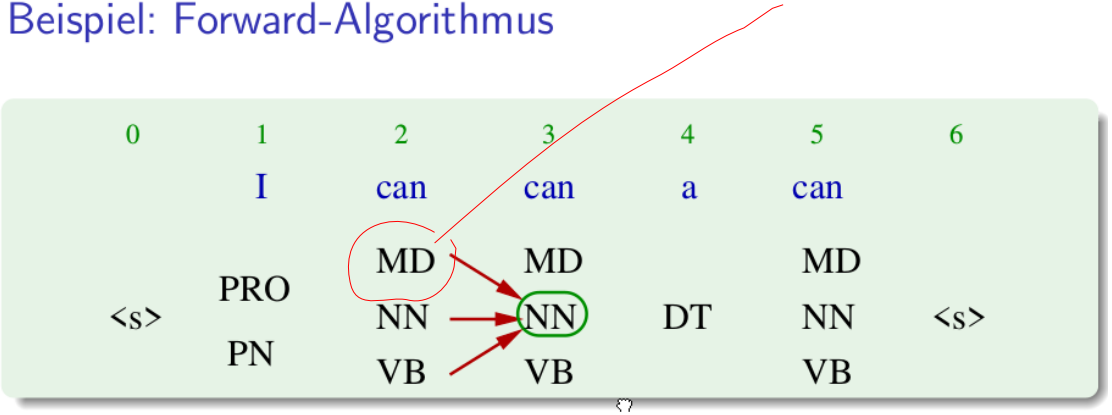
Z.B wir wollen f(NN, can) schätzen und wir sehen, dieser Satz hat das Wort “can” an Position 3, deswegen wollen wir seine **y_NN(3)** berechnen. Dafür müssen wir einige Forward und Backward-WK berechnen.

Hier die Berechnung von Forward_NN(3)

$$\gamma_t(k) = \frac{\alpha_t(k)\,\beta_t(k)}{\alpha_{\langle s\rangle}(n+1)}$$

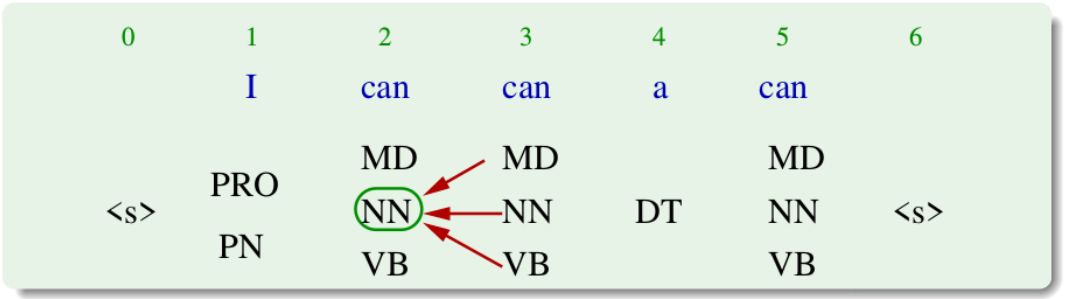
Übung

Beispiel: Forward-Algorithmus



Berechnung der Forward-Wahrscheinlichkeit des Tags NN an Position 3:

$$\begin{aligned}\alpha_{NN}(3) = {} & \alpha_{MD}(2)\; p(NN|MD)\; p(can|NN) + \\ & \alpha_{NN}(2)\; p(NN|NN)\; p(can|NN) + \\ & \alpha_{VB}(2)\; p(NN|VB)\; p(can|NN)\end{aligned}$$



Berechnung der Backward-Wahrscheinlichkeit des Tags NN an Position 2:

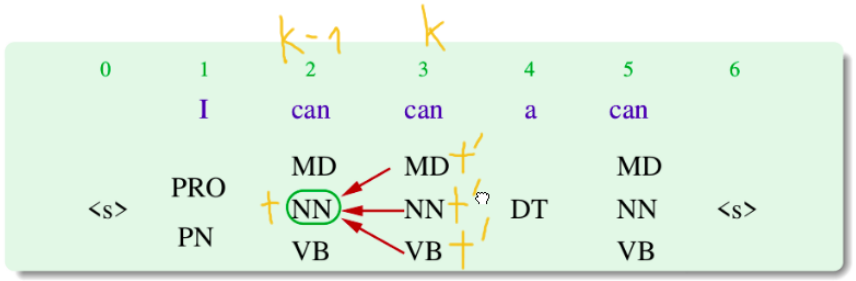
$$\begin{aligned}\beta_{NN}(2) = \; & \beta_{MD}(3)\; p(MD|NN)\; p(can|MD) + \\ & \beta_{NN}(3)\; p(NN|NN)\; p(can|NN) + \\ & \beta_{VB}(3)\; p(VB|NN)\; p(can|VB)\end{aligned}$$

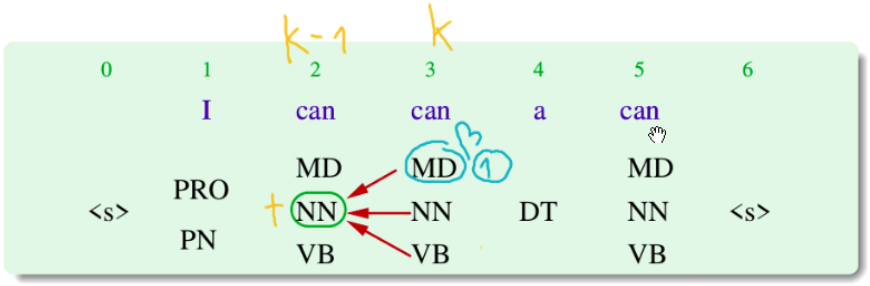
Die Backward-Wahrscheinlichkeit $\beta_t(k)$ des Tags t an Position k ist die Summe der Wahrscheinlichkeiten aller Tagfolgen t_k^{n+1} für die Teil-Wortfolge w_{k+1}^{n+1}, die mit dem Tag t beginnen und dem Tag $\langle s \rangle$ enden.

Die lexikalische Wk. $p(w_k|t_k)$ ist in $\beta_t(k)$ nicht enthalten!

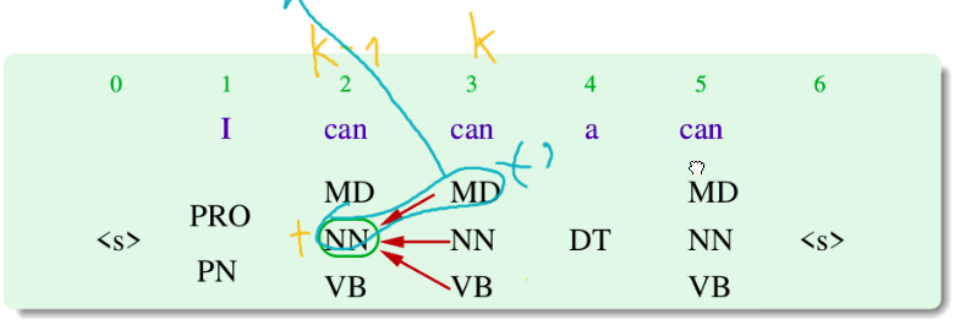
Formeln für die rekursive Berechnung im Fall des Bigramm-Taggers:

$$\begin{aligned}\beta_t(n+1) &= \begin{cases} 1 & \text{falls } t = \langle s \rangle \\ 0 & \text{sonst} \end{cases} \\ \beta_t(k-1) &= \sum_{t' \in T} p(t'|t)\; p(w_k|t')\; \beta_{t'}(k) \qquad \text{für } 0 < k \leq n+1\end{aligned}$$

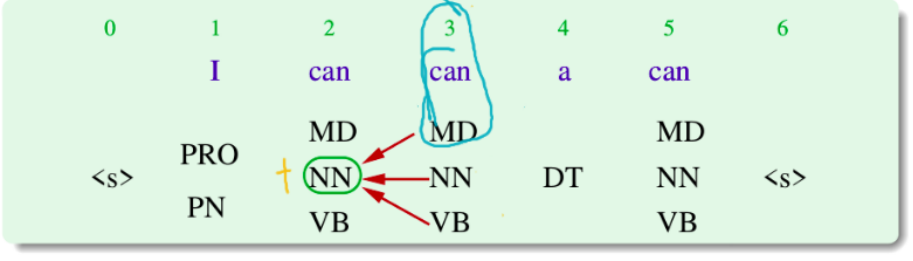




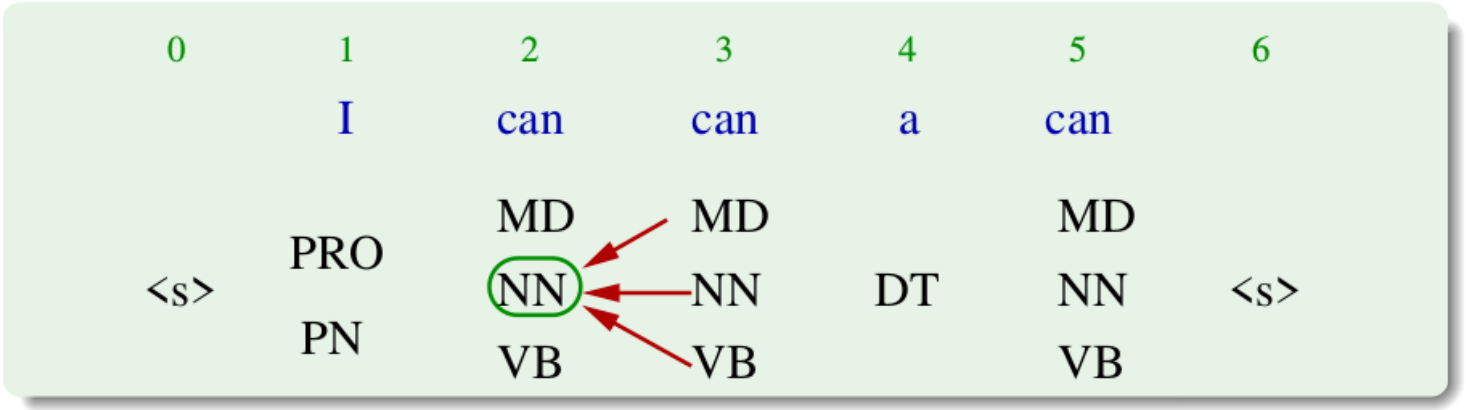
$$\beta_t(k-1) = \sum_{t' \in T} p(t'|t)\, p(w_k|t')\, \beta_{t'}(k) \quad \text{für } 0 < k \leq n+1$$



$$\beta_t(k-1) = \sum_{t' \in T} p(t'|t)\, p(w_k|t')\, \beta_{t'}(k) \quad \text{für } 0 < k \leq n+1$$



Beispiel: Backward-Algorithmus



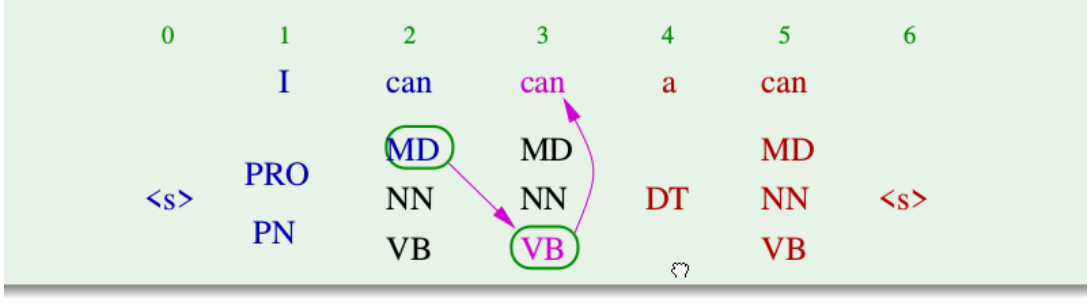
Berechnung der Backward-Wahrscheinlichkeit des Tags NN an Position 2:

$$\begin{aligned}\beta_{NN}(2) = & \beta_{MD}(3)\, p(MD|NN)\, p(can|MD) + \\ & \beta_{NN}(3)\, p(NN|NN)\, p(can|NN) + \\ & \beta_{VB}(3)\, p(VB|NN)\, p(can|VB)\end{aligned}$$

Forward-Backward-Algorithmus

0	1	2	3	4	5	6
	I	can	can	a	can	
<s>	PRO PN	MD NN VB	MD NN VB	DT	MD NN VB	<s>

$$\gamma_{NN}(3) = \frac{\alpha_{NN}(3)\,\beta_{NN}(3)}{\alpha_{\langle s\rangle}(6)}$$



$$\gamma_{MD,VB}(2) = \frac{\alpha_{MD}(2)\; p(VB|MD)\; p(can|VB)\; \beta_{VB}(3)}{\alpha_{\langle s\rangle}(6)}$$

Aufgabe 7) Wie kann ein Wortart-Tagger auf ungetaggten Daten trainiert werden? Welche Daten benötigen Sie dafür? Wie läuft das Training ab? Wie lauten die Formeln für den dabei verwendeten Algorithmus? (5 Punkte)

Aufgabe 7) Erläutern Sie den EM-Algorithmus am Beispiel des unüberwachten Trainings von PCFGs (also Training auf Rohtexten). Welche Daten benötigen Sie? Welche Berechnungsschritte führt der EM-Algorithmus aus? (4 Punkte)

Aufgabe 6) Was wird im E-Schritt und im M-Schritt des EM-Algorithmus jeweils berechnet? (Ein Satz genügt jeweils.) (1 Punkt)

Aufgabe 4) Nach welchen Formeln berechnen Sie die Forward- und Backward-Wahrscheinlichkeiten mit einem Hidden-Markow-Modell? (4 Punkte)

Wie berechnen Sie aus den Forward- und Backward-Wahrscheinlichkeiten die Wahrscheinlichkeit des Tags t an Position i bzw. die Wahrscheinlichkeit der Tags t und t' an den Positionen i-1 und i? (In der Vorlesung wurde diese Werte mit γ bezeichnet.)
(3 Punkte)

Aufgabe 8) Angenommen Sie trainieren einen HMM-Tagger mit dem Forward-Backward-Algorithmus. Wie können Sie die **erwartete Häufigkeit** (= Aposteriori-Wahrscheinlichkeit) des Tags t an der Position des Wortes w_k aus den Forward-Wahrscheinlichkeiten $\alpha_t(k)$ und Backward-Wahrscheinlichkeiten $\beta_t(k)$ berechnen?

Wie berechnen Sie die erwartete Häufigkeit des Tagpaares t und t' an den Positionen der Wörter w_k und w_{k+1}? (2 Punkte)

Aufgabe 7) Der Forward-Backward-Algorithmus wird verwendet, um erwartete Häufigkeiten von Wort-Tag-Paaren und von Tag-Tag-Paaren zu berechnen.

Geben Sie die Formeln an für die Berechnung (i) der Forward-Wahrscheinlichkeiten $\alpha_t(i)$ (ii) der Backward-Wahrscheinlichkeiten $\beta_t(i)$ und (iii) der erwarteten Häufigkeit $\gamma_t(i)$ (= Aposteriori-Wahrscheinlichkeit) des Wortes w_i mit dem Tag t an. (5 Punkte)

Aufgabe 10) Erklären Sie das EM-Training am Beispiel des unüberwachten Trainings von Wortart-Taggern. Was ist das Grundprinzip? Welche Schritte umfasst das Verfahren? Was sollte gegeben sein? Mit welchem Algorithmus kann das EM-Training bei einem HMM-Tagger effizient implementiert werden?

(3 Punkte)