note:

- The lecture slide has been updated. do not forget to download the latest version

content

- Viterbi, forward-backward algorithm, EM-training

Änderungen

Datum: 23. 10. 2021 Korrektur der Formel auf Seite 36

Datum: 3. 11. 2021 Folie 43 ergänzt, verschiedene kleinere Änderungen

Datum: 4. 11. 2021 Überarbeitung der Folien zum Thema statistische Tests

Datum: 11. 11. 2021 neue Folie zum p-Wert

Datum: 18. 11. 2021 neue einleitende Folie zur Backoff-Glättung

Datum: 6. 12. 2021 neue Folie 71 zum Endesymbol

Datum: 9. 12. 2021 Beispiel zur Pseudowort-Desambiguierung

Datum: 9. 12. 2021 Kurzzusammenfassung für verschiedene Themen

Viterbi

I	want	to	shop	for	a	new	dress
PRP	VBP	TO	VB	IN	DT	JJ	NN
PRP	VBP	TO	VB	IN	DT	JJ	VB
PRP	VBP	TO	NN	IN	DT	JJ	NN
PRP	VBP	TO	VB	IN	DT	JJ	VB

model

PRP VBP TO VB IN DT JJ NN

$$\hat{t}_1^n = \arg\max_{t_1^n} p(t_1^n|w_1^n)$$

$$p(t_1^n, w_1^n) = \prod_{i=1}^{n+1} \underbrace{p(t_i|t_{i-k}^{i-1})}_{\text{Kontextwahrsch.}} \underbrace{p(w_i|t_i)}_{\text{lexikalische Wk.}}$$

p(t1,t2,t3 | w1,w2,w3) = 0.5
p(t1,t2,t3 | w1,w2,w3) = 0.2
p(t1,t2,t3 | w1,w2,w3) = 0.2
p(t1,t2,t3 | w1,w2,w3) = 0.1

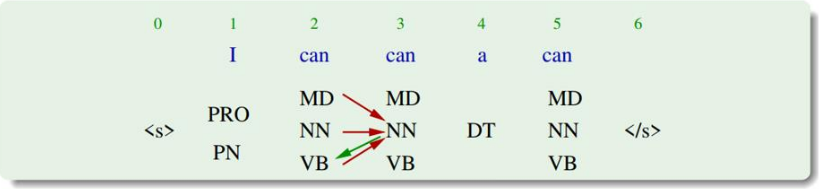
Instead of computing the best tag seq this way, we use the Viterbi algorithm instead because it is more efficient.

compute p for all possible tag seq candidates and argmax

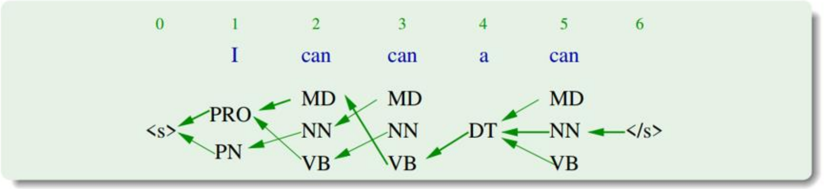
Viterbi Algorithm

With Viterbi, we only have to know the possible tags for each word but we do not have to list all possible tag sequences for the computation.

1. compute Viterbi probability for every tag at every position in the input text.



2. compute the best tag sequence



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

2. Berechnung: (für $1 \leq k \leq n+1$)

$$\delta_t(k) = \max_{t'} \delta_{t'}(k-1)\, \underline{p(t|t')\, p(w_k|t)}$$
$$\psi_t(k) = \arg\max_{t'} \delta_{t'}(k-1)\, p(t|t')\, p(w_k|t)$$

3. Ausgabe: (für $0 < k \leq n$)

$$t_{n+1} = \langle /s \rangle$$
$$t_k = \psi_{t_{k+1}}(k+1) \quad \text{für } 1 \leq k \leq n$$

To compute the Viterbi prob, we also need the lexical prob and the context prob like in the first approach.

Reason to use Viterbi

Problem: Viele Sätze haben zuviele mögliche Tagfolgen, um alle aufzählen und ihre Wahrscheinlichkeit berechnen zu können.

Wie könnte eine effizientere Methode für die Berechnung der besten Tagfolge aussehen?

Beobachtung: Wenn zwei mögliche Wortartfolgen $T = t_1, ..., t_m$ und $S = s_1, ..., s_m$ (für die ersten m Wörter eines Satzes) in den letzten k Tags übereinstimmen, dann kann bei einem HMM k-ter Ordnung die weniger wahrscheinliche der beiden Tagfolgen nicht ein Präfix der besten Gesamttagfolge sein, und wir können sie ignorieren.

Widerspruchsbeweis

Angenommen S ist weniger wahrscheinlich als T, aber ein Präfix der besten Tagfolge. Dann hat die beste Tagfolge die Form SV, wobei V eine weitere Tagfolge ist. In diesem Fall hätte aber die Tagfolge TV eine höhere Wahrscheinlichkeit als SV, weil T wahrscheinlicher als S ist und die Wahrscheinlichkeiten, die jeweils für V hinzumultipliziert werden, bei SV und TV identisch sind. Also kann S kein Präfix der besten Tagfolge sein. □

Beispiel: k=2 T=X,A,B S=Y,A,B V=C,D
TV=X,A,B,C,D SV=Y,A,B,C,D

C hängt hier nur von A und B ab. D hängt nur von B und C ab.
Also sind die Tags C,D nach X,A,B genauso wahrscheinlich wie nach Y,A,B.
Also muss die Tagfolge X,A,B,C,D wahrscheinlicher als Y,A,B,C,D sein.

Diese Eigenschaft erlaubt eine effiziente Verarbeitung, hat aber zur Folge, dass manche Ambiguitäten nicht korrekt aufgelöst werden können:

The	horse	raced	past	the	barn	(fell)	.
DT	NN	VBD	IN	DT	NN		.
DT	NN	VBN	IN	DT	NN	VBD	.

Ein HMM 2. Ordnung würde das Wort “raced” gleich taggen, egal ob “fell” nachfolgt oder nicht. Erst ein HMM 4. Ordnung kann weit genug zurückschauen, um korrekt zu desambiguieren.

Anmerkung: Man könnte das Problem lösen, indem man das Tagset verfeinert und bspw. alle Tags, die nach einem finiten Verb folgen, mit einem Apostroph markiert.

The	horse	raced	past	the	barn	(fell)	.
DT	NN	VBD	IN’	DT’	NN’		.
DT	NN	VBN	IN	DT	NN	VBD	.

Viterbi-Algorithmus (Bigramm-Tagger)

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

2. Berechnung: (für $1 \leq k \leq n+1$)

$$\begin{aligned} \delta_t(k) &= \max_{t'} \delta_{t'}(k-1)\, p(t|t')\, p(w_k|t) \\ \psi_t(k) &= \arg\max_{t'} \delta_{t'}(k-1)\, p(t|t')\, p(w_k|t) \end{aligned}$$

3. Ausgabe: (für $0 < k \leq n$)

$$\begin{aligned} t_{n+1} &= \langle /s \rangle \\ t_k &= \psi_{t_{k+1}}(k+1) \quad \text{für } 1 \leq k \leq n \end{aligned}$$

t, t' sind Tags, k sind Wortpositionen, $\delta_t(k)$ sind die Viterbiwahrscheinlichkeiten und $\psi_t(k)$ sind die besten Vorgängertags

important: you should not only remember how the algorithm works in general but **should also try to understand the formula.** If the formula is modified, you should be able to use it correctly.

For example,

- when the formula is changed to 3-gram or 4-gram
- the context and lexical prob could also be modified
- changes related to position k
- or if we use other variables instead of t or t‘

To compute the best tag seq using Viterbi, start by writing the input sentence and the tags for each word like this

1	2	3	4	5
I	can	can	a	can
PRO PN	MD NN VB	MD NN VB	DT	MD NN VB

Then add the start and end symbols depending on the n-gram order of your model

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>

Viterbi-Algorithmus (Bigramm-Tagger)

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

2. Berechnung: (für $1 \leq k \leq n+1$)

$$\delta_t(k) = \max_{t'} \delta_{t'}(k-1)\, p(t|t')\, p(w_k|t)$$

$$\psi_t(k) = \arg\max_{t'} \delta_{t'}(k-1)\, p(t|t')\, p(w_k|t)$$

3. Ausgabe: (für $0 < k \leq n$)

$$t_{n+1} = \langle /s \rangle$$

$$t_k = \psi_{t_{k+1}}(k+1) \quad \text{für } 1 \leq k \leq n$$

t, t' sind Tags, k sind Wortpositionen, $\delta_t(k)$ sind die Viterbiwahrscheinlichkeiten und $\psi_t(k)$ sind die besten Vorgängertags

Viterbi-Algorithmus (Bigramm-Tagger)

1. Initialisierung: $\delta_t(0) = \begin{cases} 1 & \text{falls } t = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$ Viterbi prob for tag t at position k=0

2. Berechnung: (für $1 \leq k \leq n+1$) for position k=1 to k=n+1

$$\begin{aligned} \delta_t(k) &= \max_{t'} \delta_{t'}(k-1)\, p(t|t')\, p(w_k|t) \\ \psi_t(k) &= \arg\max_{t'} \delta_{t'}(k-1)\, p(t|t')\, p(w_k|t) \end{aligned}$$

3. Ausgabe: (für $0 < k \leq n$)

$$\begin{aligned} t_{n+1} &= \langle /s \rangle \\ t_k &= \psi_{t_{k+1}}(k+1) \quad \text{für } 1 \leq k \leq n \end{aligned}$$

t, t' sind Tags, k sind Wortpositionen, $\delta_t(k)$ sind die Viterbiwahrscheinlichkeiten und $\psi_t(k)$ sind die besten Vorgängertags

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

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

2. Berechnung: (für $1 \leq k \leq n+1$) for position k=1 to k=n+1

$$\delta_t(k) \quad = \quad \max_{t'} \; \delta_{t'}(k-1)\; p(t|t')\; p(w_k|t)$$

0	1	2	3	4	5	6
	I	can	can	a	can	
<s>	PRO 0,075	MD 0.3	MD 0.3	DT 0.3	MD 0.1	</s> 0.45
1	PN 0.003	NN 0.5	NN 0.7		NN 0.2	
		VB 0.4	VB 0.1		VB 0.3	

At every tag, we would also record the best tag of the previous position (psi_k stores the best tag for position k-1).

$$\psi_t(k) \quad = \quad \arg\max_{t'} \; \delta_{t'}(k-1)\; p(t|t')\; p(w_k|t)$$

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

PRO is the tag (from the previous position) that leads to the maximum value of

$$\delta_{t'}(k-1)\; p(t|t')\; p(w_k|t)$$

3. Ausgabe: (für $0 < k \leq n$) for k= n to k=1

$$t_{n+1} = \langle /s \rangle$$
$$t_k = \psi_{t_{k+1}}(k+1) \quad \text{für } 1 \leq k \leq n$$

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

- Here we want to extract the best tag for each position.
- We start the computation from the back to the front.
- first, tag at n+1 is always </s>
- for k= n to k=1
  - extract the best tag at k by looking at the tag we store at k+1

NN is the best tag for position 5 (if position 6 has the tag </s>)

If the best tag at 5 is NN, then the best tag at 4 is DT

pos: 6 </s>

pos: 5 NN
pos: 4 DT
pos: 3 VB
pos: 2 VB
pos: 1 PRO

reverse

final output (return only tags for pos 1 to n)

PRO, VB, VB, DT, NN

Example: how to compute the Viterbi prob

	AUX	DT	NN	PP	VB	VBP	$\langle s\rangle$
AUX	0.01	0.1	0.2	0.2	0.01	0.01	0.01
DT	0.01	0.01	0.2	0.2	0.5	0.45	0.5
NN	0.01	0.8	0.1	0.2	0.22	0.22	0.2
PP	0.01	0.01	0.1	0.1	0.2	0.2	0.25
VB	0.94	0.01	0.01	0.01	0.01	0.01	0.02
VBP	0.01	0.06	0.29	0.19	0.01	0.01	0.01
$\langle /s\rangle$	0.01	0.01	0.1	0.1	0.05	0.1	0.01
I				0.3			
he				0.7			
can	0.5		0.3		0.1		
will	0.3		0.2				
must	0.2		0.1				
car			0.2				
house			0.2		0.1	0.2	
a		0.4					
the		0.6					
read					0.5	0.5	
write					0.3	0.3	

Die Wahrscheinlichkeit, dass das Tag VB auf das Tag AUX folgt ist hier bspw. 0.94.

0	1	2	3	4
	I	can	can	
<s>	PP	AUX	AUX	</s>
		NN	NN	
		VB	VB	

```
Viterbi-Algorithmus

d(S,0) = 1

d(PP,1) = d(S,0) p(PP|S) p(I|PP) = 1 * 0.25 * 0.3 = 0,075
psi(PP,1) = S

d(AUX,2) = d(PP,1) p(AUX|PP) p(can|AUX) = 0,075 * 0,2 * 0,5 = 0,0075
d(NN,2) = d(PP,1) p(NN|PP) p(can|NN) = 0,075 * 0,2 * 0,3 = 0,0045
d(VB,2) = d(PP,1) p(VB|PP) p(can|VB) = 0,075 * 0,01 * 0,1 = 0,000075
psi(AUX,2) = PP
psi(NN,2) = PP
psi(VB,2) = PP

d(AUX,3) = max(d(AUX,2) p(AUX|AUX) p(can|AUX),
               d(NN,2) p(AUX|NN) p(can|AUX),
               d(VB,2) p(AUX|VB) p(can|AUX))
         = max(0,0075 * 0.01 * 0.5, 0.0045 * 0.2 * 0.5, 0,000075 * 0.94 * 0.5) = max(0.0000375 , 0.00045, 0,000000375) = 0.00045

d(NN,3) = max(d(AUX,2) p(NN|AUX) p(can|NN),
              d(NN,2) p(NN|NN) p(can|NN),
              d(VB,2) p(NN|VB) p(can|NN))
         = max(0,0075 * 0.01 * 0.3, 0.0045 * 0.1 * 0.3, 0,000075 * 0.22 * 0.3) = max(0.0000225 , 0.000135, 0.00000495) = 0.000135

d(VB,3) = max(d(AUX,2) p(VB|AUX) p(can|VB),
              d(VB,2) p(VB|NN) p(can|VB),
              d(VB,2) p(VB|VB) p(can|VB))
        = max(0,0075 * 0.94 * 0.1, 0.0045 * 0.01 * 0.1, 0,000075 * 0.01 * 0.1) = max(0.000705 , 0.0000045, 0.00000075) = 0.000705
psi(AUX,3)= NN
psi(NN,3) = NN
psi(VB,3) = AUX
```

(I guess) the lexical prob is 1 when the word is empty. So, p(empty | </s>) is always 1. That is why we do not write it here.

```
d(S,4) = max(d(AUX,3) p(S|AUX), d(NN,3) p(S|NN), d(VB,3) p(S|VB))
       = max(0.00045 * 0.01, 0.000135 * 0.1, 0.000705 * 0.05)
       = max(0.0000045, 0.0000135, 0.00003525) = 0.0003525
psi(S,4) = VB

beste Tagfolge: PP AUX VB
```

0	1	2	3	4
	I	can	can	
<s>	PP	AUX	AUX	</s>
vit(<s>, 0) = 1	vit(PP,1)=0.075 psi(pp,1) = <s>	NN	NN	
		VB	VB	

Viterbi-Algorithmus (Trigramm-Tagger)

- Beim Trigramm-Tagger entspricht jeder Zustand des Hidden-Markow-Modelles nicht einem einzelnen Tag, sondern einem Tagpaar.
- Übergänge gibt es nur zwischen Zuständen (t, t') und (t'', t''') mit $t' = t''$

1. Initialisierung: $$\delta_{t',t}(0) = \begin{cases} 1 & \text{falls } t = \langle s \rangle \text{ und } t' = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$$

2. Berechnung: (für $0 < k \leq n+2$)

$$\begin{aligned} \delta_{t',t}(k) &= \max_{t''} \delta_{t'',t'}(k-1)\ p(t|t'',t')\ p(w_k|t) \\ \psi_{t',t}(k) &= \arg\max_{t''} \delta_{t'',t'}(k-1)\ p(t|t'',t')\ p(w_k|t) \end{aligned}$$

Wir fügen hier 2 Endetags hinzu und iterieren bis n+2. Das hat den Vorteil, dass wir das letzte Tag direkt in $\psi_{\langle /s \rangle,\langle /s \rangle}(n+2)$ finden. Andernfalls müssten wir erst in der Spalte n+1 durch Maximierung über t nach dem Eintrag $\delta_{t,\langle /s \rangle}(n+1)$ mit der größten Viterbi-Wahrscheinlichkeit suchen, um von dort ausgehend die beste Tagfolge zu extrahieren.

1. Initialisierung: $\delta_{t',t}(0) = \begin{cases} 1 & \text{falls } t = \langle s \rangle \text{ und } t' = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$

-1	0	1	2	3	4	5	6	7
		I	can	can	a	can		
<s>	<s>	PRO	MD	MD	DT	MD	</s>	</s>
	vit_<s><s>							
		PN	NN	NN		NN		
			VB	VB		VB		

for k=1 to k=n+2

2. Berechnung: (für $0 < k \leq n+2$)

$$\delta_{t',t}(k) \quad = \quad \max_{t''} \delta_{t'',t'}(k-1)\, p(t|t'',t')\, p(w_k|t)$$

						k=n	k=n+1	k=n+2
-1	0	1	2	3	4	5	6	7
		I	can	can	a	can		
<s>	<s>	PRO	MD	MD	DT	MD	</s>	</s>
	vit_<s><s>	vit<s>,PRO	vit_PRO, MD vit_PN, MD	vit_MD, MD vit_NN, MD vit_VB, MD	vit_MD, DT vit_NN, DT vit_VB, DT	vit_DT, MD	vit_MD, </s> vit_NN, </s> vit_VB, </s>	vit_</s>,</s>
		PN	NN	NN		NN		
		vit<s>,PN	vit_PRO,NN vit_PN,NN	vit_MD, NN vit_NN, NN vit_VB, NN		vit_DT, NN		
			VB	VB		VB		
			vit_PRO,VB vit_PN,VB	vit_MD, VB vit_NN, VB vit_VB, VB		vit_DT, VB		

2. Berechnung: (für $0 < k \leq n + 2$)

$$\delta_{t',t}(k) \quad = \quad \max_{t''} \; \delta_{t'',t'}(k-1)\; p(t|t'',t')\; p(w_k|t)$$

			k=2					
	t''	t'	t					
-1	0	1	2	3	4	5	6	7
		I	can	can	a	can		
<s>	<s> vit_<s><s>	PRO vit<s>,PRO	MD vit_PRO, MD vit_PN, MD	MD vit_MD, MD vit_NN, MD vit_VB, MD	DT vit_MD, DT vit_NN, DT vit_VB, DT	MD vit_DT, MD	</s> vit_MD, </s> vit_NN, </s> vit_VB, </s>	</s> vit_</s>,</s>
		PN vit<s>,PN	NN vit_PRO,NN vit_PN,NN	NN vit_MD, NN vit_NN, NN vit_VB, NN		NN vit_DT, NN		
			VB vit_PRO,VB vit_PN,VB	VB vit_MD, VB vit_NN, VB vit_VB, VB		VB vit_DT, VB		

$$\delta_{\langle s\rangle,\langle s\rangle}(0) = 1$$

$$\delta_{\langle s\rangle,PRO}(1) = \delta_{\langle s\rangle,\langle s\rangle}(0)\ \mathrm{p}(\mathrm{PRO}|\langle s\rangle,\langle s\rangle)\ \mathrm{p}(\mathrm{I}|\mathrm{PRO})$$
$$\delta_{\langle s\rangle,PN}(1) = \delta_{\langle s\rangle,\langle s\rangle}(0)\ \mathrm{p}(\mathrm{PN}|\langle s\rangle,\langle s\rangle)\ \mathrm{p}(\mathrm{I}|\mathrm{PN})$$

$$\delta_{PRO,MD}(2) = \delta_{\langle s\rangle,PRO}(1)\ \mathrm{p}(\mathrm{MD}|\langle s\rangle,\mathrm{PRO})\ \mathrm{p}(\mathrm{can}|\mathrm{MD})$$
$$\delta_{PRO,NN}(2) = \delta_{\langle s\rangle,PRO}(1)\ \mathrm{p}(\mathrm{NN}|\langle s\rangle,\mathrm{PRO})\ \mathrm{p}(\mathrm{can}|\mathrm{NN})$$
$$\delta_{PRO,VB}(2) = \delta_{\langle s\rangle,PRO}(1)\ \mathrm{p}(\mathrm{VB}|\langle s\rangle,\mathrm{PRO})\ \mathrm{p}(\mathrm{can}|\mathrm{VB})$$

$$\delta_{PN,MD}(2) = \delta_{\langle s\rangle,PN}(1)\ \mathrm{p}(\mathrm{MD}|\langle s\rangle,\mathrm{PN})\ \mathrm{p}(\mathrm{can}|\mathrm{MD})$$
$$\delta_{PN,NN}(2) = \delta_{\langle s\rangle,PN}(1)\ \mathrm{p}(\mathrm{NN}|\langle s\rangle,\mathrm{PN})\ \mathrm{p}(\mathrm{can}|\mathrm{NN})$$
$$\delta_{PN,VB}(2) = \delta_{\langle s\rangle,PN}(1)\ \mathrm{p}(\mathrm{VB}|\langle s\rangle,\mathrm{PN})\ \mathrm{p}(\mathrm{can}|\mathrm{VB})$$

$$\delta_{MD,MD}(3) = \max(\delta_{PRO,MD}(2)\ \mathrm{p}(\mathrm{MD}|\mathrm{PRO},\mathrm{MD})\ \mathrm{p}(\mathrm{can}|\mathrm{MD}),\ \delta_{PN,MD}(2)\ \mathrm{p}(\mathrm{MD}|\mathrm{PN},\mathrm{MD})\ \mathrm{p}(\mathrm{can}|\mathrm{MD}))$$
$$\delta_{MD,NN}(3) = \max(\delta_{PRO,MD}(2)\ \mathrm{p}(\mathrm{NN}|\mathrm{PRO},\mathrm{MD})\ \mathrm{p}(\mathrm{can}|\mathrm{NN}),\ \delta_{PN,MD}(2)\ \mathrm{p}(\mathrm{NN}|\mathrm{PN},\mathrm{MD})\ \mathrm{p}(\mathrm{can}|\mathrm{NN}))$$
$$\delta_{MD,VB}(3) = \max(\delta_{PRO,MD}(2)\ \mathrm{p}(\mathrm{VB}|\mathrm{PRO},\mathrm{MD})\ \mathrm{p}(\mathrm{can}|\mathrm{VB}),\ \delta_{PN,MD}(2)\ \mathrm{p}(\mathrm{VB}|\mathrm{PN},\mathrm{MD})\ \mathrm{p}(\mathrm{can}|\mathrm{VB}))$$

$$\delta_{NN,MD}(3) = \max(\delta_{PRO,NN}(2)\ \mathrm{p}(\mathrm{MD}|\mathrm{PRO},\mathrm{NN})\ \mathrm{p}(\mathrm{can}|\mathrm{MD}),\ \delta_{PN,NN}(2)\ \mathrm{p}(\mathrm{MD}|\mathrm{PN},\mathrm{NN})\ \mathrm{p}(\mathrm{can}|\mathrm{MD}))$$

....

2. Berechnung: (für $0 < k \leq n + 2$)   for k=1 to k=n+2

$$\delta_{t',t}(k) = \max_{t''} \delta_{t'',t'}(k-1)\ p(t|t'',t')\ p(w_k|t)$$

$$\psi_{t',t}(k) = \arg\max_{t''} \delta_{t'',t'}(k-1)\ p(t|t'',t')\ p(w_k|t)$$

psi stores the best tag for position k-2 (t‘‘)

-1	0	1	2	3	4	5	6	7
		I	can	can	a	can		
<s>	<s>	PRO psi_<s>,PRO= <s>	MD psi_PRO,MD= <s> psi_PN,MD= <s>	MD psi_MD,MD= PRO psi_NN,MD= PN psi_VB,MD= PN	DT psi_MD,DT= MD psi_NN,DT= VB psi_VB,DT= NN	MD psi_DT,MD = VB	</s> psi_MD,</s> = DT psi_NN,</s> = DT psi_VB,</s> = DT	</s> psi_</s>,</s> = NN
		PN psi_<s>,PN= <s>	NN psi_PRO,NN= <s> psi_PN,NN= <s>	NN psi_MD,NN= PN psi_NN,NN= PRO psi_VB,NN= PRO		NN psi_DT,NN = VB		
			VB psi_PRO,VB= <s> psi_PN,VB= <s>	VB psi_MD,VB= PRO psi_NN,VB= PRO psi_VB,VB= PN		VB psi_DT,VB = NN		

extract the best tag sequence

for k=n to k= 1

3. Ausgabe: (für $0 < k \le n$)

$$t_{n+1} = \langle/s\rangle, \quad t_{n+2} = \langle/s\rangle$$

$$t_k = \psi_{t_{k+2}}(k+2)$$

- We will start extracting the best tag at k=n (pos 5) until k=1 (pos 1)
- Before we do that we first have to set the best tag at pos 6 and 7 to be </s> (marked with green)

-1	0	1	2	3	4	5	6	7
		I	can	can	a	can		
<s>	<s>	PRO psi_<s>,PRO= <s>	MD psi_PRO,MD= <s> psi_PN,MD= <s>	MD psi_MD,MD= PRO psi_NN,MD= PN psi_VB,MD= PN	DT psi_MD,DT= MD psi_NN,DT= VB psi_VB,DT= NN	MD psi_DT,MD = VB	</s> psi_MD,</s> = DT psi_NN,</s> = DT psi_VB,</s> = DT	</s> psi_</s>,</s> = NN
		PN psi_<s>,PN= <s>	NN psi_PRO,NN= <s> psi_PN,NN= <s>	NN psi_MD,NN= PN psi_NN,NN= PRO psi_VB,NN= PRO		NN psi_DT,NN = VB		
			VB psi_PRO,VB= <s> psi_PN,VB= <s>	VB psi_MD,VB= PRO psi_NN,VB= PRO psi_VB,VB= PN		VB psi_DT,VB = NN		

pos	1	2	3	4	5
output					

extract the best tag sequence

what is the best tag at pos 5?

- look at psi at pos 7
- it is NN

-1	0	1	2	3	4	5	6	7
		I	can	can	a	can		
<s>	<s>	PRO psi_<s>,PRO= <s>	MD psi_PRO,MD= <s> psi_PN,MD= <s>	MD psi_MD,MD= PRO psi_NN,MD= PN psi_VB,MD= PN	DT psi_MD,DT= MD psi_NN,DT= VB psi_VB,DT= NN	MD psi_DT,MD = VB	</s> psi_MD,</s> = DT psi_NN,</s> = DT psi_VB,</s> = DT	</s> psi_</s>,</s> = NN
		PN psi_<s>,PN= <s>	NN psi_PRO,NN= <s> psi_PN,NN= <s>	NN psi_MD,NN= PN psi_NN,NN= PRO psi_VB,NN= PRO		NN psi_DT,NN = VB		
			VB psi_PRO,VB= <s> psi_PN,VB= <s>	VB psi_MD,VB= PRO psi_NN,VB= PRO psi_VB,VB= PN		VB psi_DT,VB = NN		

pos	1	2	3	4	5
output					NN

extract the best tag sequence

what is the best tag at pos 4?

- look at psi at pos 6. Here we pick psi_NN,</s> because the best tag at position 5 is NN, and the best tag at 6 is </s>
- it is DT

-1	0	1	2	3	4	5	6	7
		I	can	can	a	can		
<s>	<s>	PRO	MD	MD	DT	MD	</s>	</s>
		psi_<s>,PRO= <s>	psi_PRO,MD= <s> psi_PN,MD= <s>	psi_MD,MD= PRO psi_NN,MD= PN psi_VB,MD= PN	psi_MD,DT= MD psi_NN,DT= VB psi_VB,DT= NN	psi_DT,MD = VB	psi_MD,</s> = DT psi_NN,</s> = DT psi_VB,</s> = DT	psi_</s>,</s> = NN
		PN	NN	NN		NN		
		psi_<s>,PN= <s>	psi_PRO,NN= <s> psi_PN,NN= <s>	psi_MD,NN= PN psi_NN,NN= PRO psi_VB,NN= PRO		psi_DT,NN = VB		
			VB	VB		VB		
			psi_PRO,VB= <s> psi_PN,VB= <s>	psi_MD,VB= PRO psi_NN,VB= PRO psi_VB,VB= PN		psi_DT,VB = NN		

pos	1	2	3	4	5
output				DT	NN

extract the best tag sequence

what is the best tag at pos 3?

- look at psi at pos 5. Here we pick psi_DT,NN because the best tag at position 4 is DT and for pos 5 it is NN
- it is VB

-1	0	1	2	3	4	5	6	7
		I	can	can	a	can		
<s>	<s>	PRO psi_<s>,PRO= <s>	MD psi_PRO,MD= <s> psi_PN,MD= <s>	MD psi_MD,MD= PRO psi_NN,MD= PN psi_VB,MD= PN	DT psi_MD,DT= MD psi_NN,DT= VB psi_VB,DT= NN	MD psi_DT,MD = VB	</s> psi_MD,</s> = DT psi_NN,</s> = DT psi_VB,</s> = DT	</s> psi_</s>,</s> = NN
		PN psi_<s>,PN= <s>	NN psi_PRO,NN= <s> psi_PN,NN= <s>	NN psi_MD,NN= PN psi_NN,NN= PRO psi_VB,NN= PRO		NN psi_DT,NN = VB		
			VB psi_PRO,VB= <s> psi_PN,VB= <s>	VB psi_MD,VB= PRO psi_NN,VB= PRO psi_VB,VB= PN		VB psi_DT,VB = NN		

pos	1	2	3	4	5
output			VB	DT	NN

extract the best tag sequence

what is the best tag at pos 2?

- look at psi at pos 4. Here we pick psi_VB,DT because the best tag at position 3 is VB and for pos 5 it is DT
- it is MD

-1	0	1	2	3	4	5	6	7
		I	can	can	a	can		
<s>	<s>	PRO psi_<s>,PRO= <s>	MD psi_PRO,MD= <s> psi_PN,MD= <s>	MD psi_MD,MD= PRO psi_NN,MD= PN psi_VB,MD= PN	DT psi_MD,DT= MD psi_NN,DT= VB psi_VB,DT= MD	MD psi_DT,MD = VB	</s> psi_MD,</s> = DT psi_NN,</s> = DT psi_VB,</s> = DT	</s> psi_</s>,</s> = NN
		PN psi_<s>,PN= <s>	NN psi_PRO,NN= <s> psi_PN,NN= <s>	NN psi_MD,NN= PN psi_NN,NN= PRO psi_VB,NN= PRO		NN psi_DT,NN = VB		
			VB psi_PRO,VB= <s> psi_PN,VB= <s>	VB psi_MD,VB= PRO psi_NN,VB= PRO psi_VB,VB= PN		VB psi_DT,VB = NN		

pos	1	2	3	4	5
output		MD	VB	DT	NN

what is the best tag at pos 1?

- look at psi at pos 3. Here we pick psi_MD,VB because the best tag at position 2 is MD and for pos 3 it is VB
- it is PRO

-1	0	1	2	3	4	5	6	7
		I	can	can	a	can		
<s>	<s>	PRO psi_<s>,PRO= <s>	MD psi_PRO,MD= <s> psi_PN,MD= <s>	MD psi_MD,MD= PRO psi_NN,MD= PN psi_VB,MD= PN	DT psi_MD,DT= MD psi_NN,DT= VB psi_VB,DT= MD	MD psi_DT,MD = VB	</s> psi_MD,</s> = DT psi_NN,</s> = DT psi_VB,</s> = DT	</s> psi_</s>,</s> = NN
		PN psi_<s>,PN= <s>	NN psi_PRO,NN= <s> psi_PN,NN= <s>	NN psi_MD,NN= PN psi_NN,NN= PRO psi_VB,NN= PRO		NN psi_DT,NN = VB		
			VB psi_PRO,VB= <s> psi_PN,VB= <s>	VB psi_MD,VB= PRO psi_NN,VB= PRO psi_VB,VB= PN		VB psi_DT,VB = NN		

finished

pos	1	2	3	4	5
output	PRO	MD	VB	DT	NN

Viterbi-Algorithmus (Trigramm-Tagger)

- Beim Trigramm-Tagger entspricht jeder Zustand des Hidden-Markow-Modelles nicht einem einzelnen Tag, sondern einem Tagpaar.
- Übergänge gibt es nur zwischen Zuständen (t, t') und (t'', t''') mit $t' = t''$

1. Initialisierung: $\delta_{t',t}(0) = \begin{cases} 1 & \text{falls } t = \langle s \rangle \text{ und } t' = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$

2. Berechnung: (für $0 < k \leq n + 2$)

$$\begin{aligned} \delta_{t',t}(k) &= \max_{t''} \delta_{t'',t'}(k-1)\, p(t|t'', t')\, p(w_k|t) \\ \psi_{t',t}(k) &= \arg\max_{t''} \delta_{t'',t'}(k-1)\, p(t|t'', t')\, p(w_k|t) \end{aligned}$$

Wir fügen hier 2 Endetags hinzu und iterieren bis n+2. Das hat den Vorteil, dass wir das letzte Tag direkt in $\psi_{\langle /s \rangle, \langle /s \rangle}(n+2)$ finden. Andernfalls müssten wir erst in der Spalte n+1 durch Maximierung über t nach dem Eintrag $\delta_{t,\langle /s \rangle}(n+1)$ mit der größten Viterbi-Wahrscheinlichkeit suchen, um von dort ausgehend die beste Tagfolge zu extrahieren.

In der Musterlösung wird die zweite Methode verwendet. (Maximierung an Position n+1)

Code example for 3-gram tagger

```
def viterbi(self, words):
    ''' berechnet die beste Tagfolge für eine gegebene Wortfolge '''
    words =  [''] + words + ['']  # Grenztokens hinzufügen

    # Initialisierung der Viterbi-Tabelle
    vitscore = [dict() for _ in range(len(words))]  # speichert logarithmierte Werte
    bestprev = [dict() for _ in range(len(words))]  # speichert die besten Vorgänger
    vitscore[0][('<s>','<s>')] = 0.0  # =log(1)
    for i in range(1, len(words)):
        lexprobs = self._lex_probs(words[i])  # die möglichen Tags nachschlagen
        for tag, lexprob in lexprobs:
            for tagpair in vitscore[i-1]:
                tag1, tag2 = tagpair  # Kontext-Tags
                p = self._context_prob(tagpair, tag) * lexprob / self._apriori_tag_prob[tag]
                p = vitscore[i-1][tagpair] + log(p)
                newtagpair = (tag2, tag)
                if newtagpair not in vitscore[i] or vitscore[i][newtagpair] < p:
                    vitscore[i][newtagpair] = p
                    bestprev[i][newtagpair] = tagpair
    # in der letzten Spalte das Tagpaar mit der höchsten Bewertung suchen
    tagpair = max(vitscore[-1], key=vitscore[-1].get)
    # beste Tagfolge extrahieren
    result_tags = []
    for i in range(len(words)-1, 1, -1):
        result_tags.append(tagpair[0])
        tagpair = bestprev[i][tagpair]
    return reversed(result_tags)  # Tagfolge umdrehen
```

Here we add only one end symbol. In the formula, we added 2

In the formula, psi saves only one tag (t''). Here we save (t'', t').

max at n+1. In the formula, we compute until n+2.

output tag t''

note: code is a bit different than the formula

EM-Training

Unüberwachtes Training

Können wir ein HMM auch ohne manuell annotierte Trainingsdaten trainieren?

- Wenn wir ein annotiertes Korpus haben, können wir ein HMM trainieren.
- Wenn wir ein trainiertes HMM haben, können wir ein Korpus annotieren.

⇒ Henne-Ei-Problem

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. Annotiere das Trainingskorpus mit dem HMM 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)

EM-Training: Beispiel

Sätze: *eine Katze jagt eine Maus* | *die entkommt der Katze* | *der Hund bellt*

context	prediction	f	p	f	p	f	p	f	p	f	p	f	p
ART	eine	1.14	0.49	1.91	0.62	2.0	0.62	2.0	0.56	2.0	0.52	2.0	0.5
VVFIN	eine	0.86	0.22	0.09	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PDS	die	0.6	0.33	0.77	0.42	0.96	0.55	1.0	0.7	1.0	0.87	1.0	0.98
ART	die	0.4	0.17	0.23	0.08	0.04	0.01	0.0	0.0	0.0	0.0	0.0	0.0
PDS	der	1.2	0.67	1.07	0.58	0.79	0.45	0.43	0.3	0.15	0.13	0.02	0.02
ART	der	0.8	0.34	0.93	0.3	1.21	0.37	1.57	0.44	1.85	0.48	1.98	0.5
VVFIN	jagt	1.0	0.26	1.0	0.32	1.0	0.33	1.0	0.33	1.0	0.33	1.0	0.33
VVFIN	entkommt	1.0	0.26	1.0	0.32	1.0	0.33	1.0	0.33	1.0	0.33	1.0	0.33
VVFIN	bellt	1.0	0.26	1.0	0.32	1.0	0.33	1.0	0.33	1.0	0.33	1.0	0.33
NN	Katze	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5
NN	Maus	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25
NN	Hund	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25
$\langle s \rangle$	ART	1.37	0.46	1.61	0.54	1.53	0.51	1.65	0.55	1.86	0.62	1.98	0.66
$\langle s \rangle$	PDS	1.2	0.4	1.35	0.45	1.47	0.49	1.35	0.45	1.14	0.38	1.02	0.34
$\langle s \rangle$	VVFIN	0.43	0.14	0.04	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ART	NN	1.94	0.83	2.84	0.92	3.21	0.99	3.57	1.0	3.85	1.0	3.98	1.0
ART	VVFIN	0.4	0.17	0.23	0.08	0.04	0.01	0.0	0.0	0.0	0.0	0.0	0.0
VVFIN	$\langle /s \rangle$	1.0	0.26	1.0	0.32	1.0	0.33	1.0	0.33	1.0	0.33	1.0	0.33
VVFIN	ART	0.97	0.25	1.46	0.47	1.71	0.57	1.91	0.64	1.99	0.66	2.0	0.67
VVFIN	NN	0.86	0.22	0.09	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
VVFIN	PDS	0.6	0.16	0.49	0.16	0.29	0.1	0.09	0.03	0.01	0.0	0.0	0.0
VVFIN	VVFIN	0.43	0.11	0.05	0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NN	VVFIN	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5
NN	$\langle /s \rangle$	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5
PDS	NN	1.2	0.67	1.07	0.58	0.79	0.45	0.43	0.3	0.15	0.13	0.02	0.02
PDS	VVFIN	0.6	0.33	0.77	0.42	0.96	0.55	1.0	0.7	1.0	0.87	1.0	0.98

Das Modell wurde wie früher beschrieben uniform initialisiert. Dann wurden die Häufigkeiten f mit dem Forward-Backward-Algorithmus geschätzt. Aus den Häufigkeiten wurden die neuen Wahrscheinlichkeiten p des HMM geschätzt. Dann wurde iteriert.

EM-Training

eine Katze jagt eine Maus | die entkommt der Katze | der Hund bellt

Lösung: Expectation-Maximization-Training

(maximiert iterativ die Wahrscheinlichkeit der Trainingsdaten)

1. gegeben
   - ein nicht annotiertes Trainingskorpus
   - ein Lexikon mit möglichen Wortarten von Wörtern

eine: DT
Katze: NN
jagt: VB, MD

---

2. Initialisiere das HMM uniform (abgesehen davon, dass $p(w|t) = 0$ falls w im Lexikon enthalten und t keines seiner möglichen Tags ist)

context	prediction
ART	eine
VVFIN	eine
PDS	die
ART	die
PDS	der
ART	der
VVFIN	jagt
VVFIN	entkommt
VVFIN	bellt
NN	Katze
NN	Maus
NN	Hund
$\langle s \rangle$	ART
$\langle s \rangle$	PDS
$\langle s \rangle$	VVFIN
ART	NN
ART	VVFIN
VVFIN	$\langle /s \rangle$
VVFIN	ART
VVFIN	NN
VVFIN	PDS

p(word| tag) = 1/N

N= number of all f(word, tag)

Now we have the lexical p and context p. We can theoretically use HMM model to annotate the corpus and extract the new freq. But p is not the real p, so it won‘t work well.

p(tag2| tag1) = 1/N

N= number of all f(tag1, tag2)

note: I‘m not sure how exactly the uniform prob is computed (I‘ll give an update once I know more about it). It is not very important, by the way.

We won't use Viterbi to annotate the sentences in the corpus then extract the frequency but we will estimate the **expected frequency directly** using another method.

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.

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.

Annotiere das Trainingskorpus mit dem HMM und extrahiere die Taghäufigkeiten (E-Schritt)

Here we won't use Viterbi to annotate the corpus, but use forward-backward probability to compute the frequency directly.

This is how the tag-word freq and tag-tag freq are computed

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

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

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

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**.

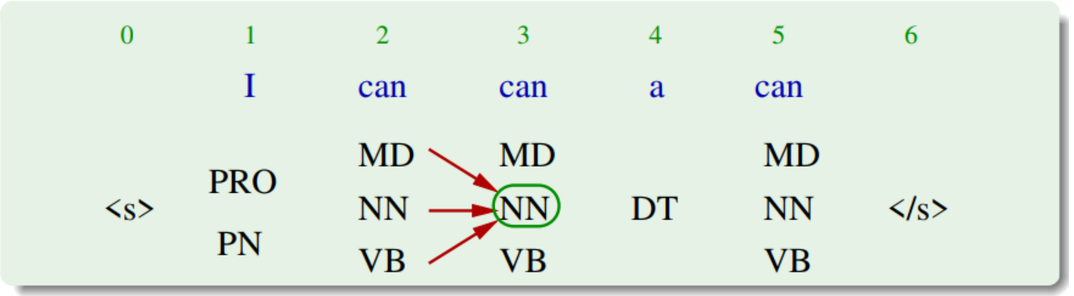
forward prob

backward prob

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

$$\gamma_{tt'}(k) = \frac{\sum_{t_1^n:t_{k-1}=t,t_k=t'} p(t_1^n,w_1^n)}{\sum_{t_1^n} p(t_1^n,w_1^n)} = \frac{\alpha_t(k-1)\ p(t'|t)\ p(w_k|t')\ \beta_{t'}(k)}{\alpha_{\langle/s\rangle}(n+1)}$$

Beispiel: Forward-Algorithmus



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

$$\alpha_{NN}(3) = \begin{array}{l} \alpha_{MD}(2)\ p(NN|MD)\ p(can|NN)+ \\ \alpha_{NN}(2)\ p(NN|NN)\ p(can|NN)+ \\ \alpha_{VB}(2)\ p(NN|VB)\ p(can|NN) \end{array}$$

Die Forward-Wahrscheinlichkeit $\alpha_t(k)$ des Tags t an Position k ist die Summe der Wahrscheinlichkeiten aller Tagfolgen t_0^k für die (Teil-)Wortfolge w_1^k, die mit dem Tag $\langle s \rangle$ beginnen und dem Tag t enden.

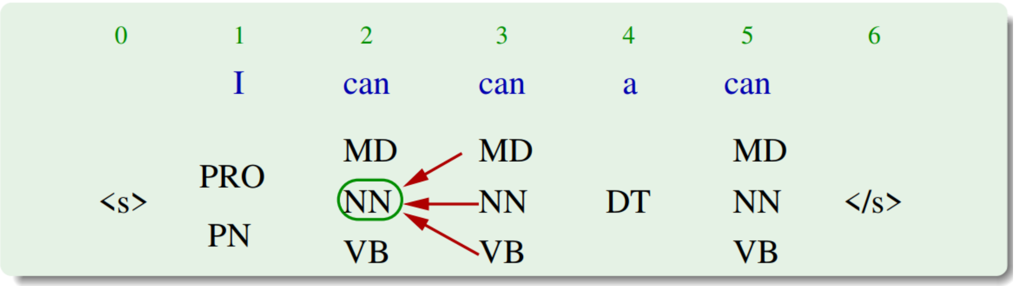
Formeln für die rekursive Berechnung im Fall des Bigramm-Taggers:

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

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

Der Forward-Algorithmus unterscheidet sich vom Viterbi-Algorithmus durch die Summe statt der Max-Operation.

Beispiel: Backward-Algorithmus



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

$$\beta_{NN}(2) = \begin{array}{l} \beta_{MD}(3)\ p(MD|NN)\ p(can|MD)+ \\ \beta_{NN}(3)\ p(NN|NN)\ p(can|NN)+ \\ \beta_{VB}(3)\ p(VB|NN)\ p(can|VB) \end{array}$$

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, 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:

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

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

Die (Aposteriori-)Wahrscheinlichkeit $p(t_k = t|w_1^n) = \gamma_t(k)$ der Wortart t an Position k

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

Analog wird die Aposteriori-Wahrscheinlichkeit

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

des Tagpaares t, t' an Position k berechnet:

$$\gamma_{tt'}(k) = \frac{\sum_{t_1^n: t_{k-1}=t, t_k=t'} p(t_1^n, w_1^n)}{\sum_{t_1^n} p(t_1^n, w_1^n)} = \frac{\alpha_t(k-1)\ p(t'|t)\ p(w_k|t')\ \beta_{t'}(k)}{\alpha_{\langle/s\rangle}(n+1)}$$

Forward-Backward-Algorithmus

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

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

$\alpha_{NN}(3)$: Wahrscheinlichkeit, dass die Wörter "I can can" mit beliebigen Tags generiert werden, wobei das 3. Tag jedoch "NN" ist.

$\beta_{NN}(3)$: Wahrscheinlichkeit, dass die Wörter "a can" mit beliebigen Tags generiert werden, falls das 3. Tag "NN" ist. (Das Endetag wird ebenfalls generiert.)

$\alpha_{\langle/s\rangle}(6)$: Forward-Wahrscheinlichkeit des Ende-Tags und damit die Gesamtwk. des Satzes summiert über alle möglichen Tagfolgen.

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

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

4. Schätze die HMM-Parameter aus den Taghäufigkeiten neu (M-Schritt)

after we have the expected freqs, we can use these freqs to compute the new p

Neuschätzung der HMM-Parameter (M-Schritt)

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

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

5. weiter mit E-Schritt (bis irgendein Abbruchkriterium erfüllt ist)

Sätze: *eine Katze jagt eine Maus* | *die entkommt der Katze* | *der Hund bellt*

context	prediction	f	p	f	p	f	p	f	p	f	p	f	p
ART	eine	1.14	0.49	1.91	0.62	2.0	0.62	2.0	0.56	2.0	0.52	2.0	0.5
VVFIN	eine	0.86	0.22	0.09	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PDS	die	0.6	0.33	0.77	0.42	0.96	0.55	1.0	0.7	1.0	0.87	1.0	0.98
ART	die	0.4	0.17	0.23	0.08	0.04	0.01	0.0	0.0	0.0	0.0	0.0	0.0
PDS	der	1.2	0.67	1.07	0.58	0.79	0.45	0.43	0.3	0.15	0.13	0.02	0.02
ART	der	0.8	0.34	0.93	0.3	1.21	0.37	1.57	0.44	1.85	0.48	1.98	0.5
VVFIN	jagt	1.0	0.26	1.0	0.32	1.0	0.33	1.0	0.33	1.0	0.33	1.0	0.33
VVFIN	entkommt	1.0	0.26	1.0	0.32	1.0	0.33	1.0	0.33	1.0	0.33	1.0	0.33
VVFIN	bellt	1.0	0.26	1.0	0.32	1.0	0.33	1.0	0.33	1.0	0.33	1.0	0.33
NN	Katze	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5	2.0	0.5
NN	Maus	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25
NN	Hund	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25	1.0	0.25
$\langle s \rangle$	ART	1.37	0.46	1.61	0.54	1.53	0.51	1.65	0.55	1.86	0.62	1.98	0.66
$\langle s \rangle$	PDS	1.2	0.4	1.35	0.45	1.47	0.49	1.35	0.45	1.14	0.38	1.02	0.34
$\langle s \rangle$	VVFIN	0.43	0.14	0.04	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ART	NN	1.94	0.83	2.84	0.92	3.21	0.99	3.57	1.0	3.85	1.0	3.98	1.0
ART	VVFIN	0.4	0.17	0.23	0.08	0.04	0.01	0.0	0.0	0.0	0.0	0.0	0.0
VVFIN	$\langle /s \rangle$	1.0	0.26	1.0	0.32	1.0	0.33	1.0	0.33	1.0	0.33	1.0	0.33
VVFIN	ART	0.97	0.25	1.46	0.47	1.71	0.57	1.91	0.64	1.99	0.66	2.0	0.67

Explain this

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

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

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

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: $\sum_{1 \leq k \leq n} \gamma_t(k) \mathbb{1}_{w_k = w}$.

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

Suppose, there are 2 sentences in corpus C. We would compute the forward and backward probability for both sentences (every tag at every position has a probability) then we will compute the Aposteriori gamma_t(k) and gamma_t,t'(k) for every tag t and position k.

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

gamma_MD(k=3)
gamma_MD,MD(k=3)

0	1	2	3	4	5	6
	She	has	a	small	baby	
<s>	PRO	MD	DT	ADV	NN	</s>
	PN	VB				

gamma_ADV(k=4)

gamma_DT,ADV(k=4)

Explain this

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

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

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

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: $\sum_{1 \leq k \leq n} \gamma_t(k) \mathbb{1}_{w_k = w}$.

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

for each sentence in the corpus and for each position k in the sentence where the word at that position = w, sum up gamma_t(k).

Example: we want to compute f(MD, can)

For each sentence, look for positions that have the word “can”.

In sentence 1, we found position2,3,5. In sentece2, we found position 5. So,

f(MD, can) = gamma_MD(k=2, sent1) +
gamma_MD(k=3, sent1) +
gamma_MD(k=5, sent1) +
gamma_MD(k=5, sent2)

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

gamma_MD(k=3)

0	1	2	3	4	5	6
	She	has	a	small	can	
<s>	PRO	MD	DT	ADV	MD	</s>
	PN	VB				

gamma_MD(k=5)

Explain this

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

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

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

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

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

for each sentence in the corpus and for each position k=1 to k=n+1, sum up gamma_tt'(k)

Example: we want to compute f(MD, DT)

For each sentence, look for positions that have gamma_MD,DT and sum them up.

f(MD, DT) = gamma_MD,DT(k=4, sent1) + gamma_MD, DT(k=3, sent2)

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

gamma_MD,DT(k=4)

0	1	2	3	4	5	6
	She	has	a	small	can	
<s>	PRO	MD	DT	ADV	DT	</s>
	PN	VB				

gamma_MD,DT(k=3)