Erklärung zur Musterlösung für Übung-SS20-WH

Trigramm-Backward-Wk

Definition der Backward-Wahrscheinlichkeiten $\beta_{t'',t'}(k)$:

$$\beta_{t'',t'}(n+2) = \begin{cases} 1 & \text{falls } t'' = t' = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$$

$$\beta_{t'',t'}(k) = \sum_{t \in T} p(t|t'',t')\; p(w_{k+1}|t)\; \beta_{t',t}(k+1) \qquad \text{für } 0 \le k \le n+1$$

	0	1	2	3	4	5	6	7
“ “	“ “	I	can	can	a	can	“ “	“ “
<s>	<s>	PRO D N V	PRO D N V	PRO D N V	PRO D N V	PRO D N V	<s>	<s>

Trigramm-Backward-Wk

Definition der Backward-Wahrscheinlichkeiten $\beta_{t'',t'}(k)$:

$$\beta_{t'',t'}(n+2) = \begin{cases} 1 & \text{falls } t'' = t' = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$$

$$\beta_{t'',t'}(k) = \sum_{t \in T} p(t|t'',t')\; p(w_{k+1}|t)\; \beta_{t',t}(k+1) \qquad \text{für } 0 \le k \le n+1$$

Um die Formel besser zu verstehen betrachten wir ein Beispiel. Z.B. wir wollen B(PRO,D, 3) berechnen, bestimmen wir die Variable t'', t', t, k, k+1, w_k+1 in der Darstellung. Wenn man p(t|t'', t') anschaut, sieht man gut welche Tag kommt in welcher Reihenfolge, also t'' dann t', dann t.

				k	k+1			
	0	1	2	3	4	5	6	7
" "	" "	I	can	can	a w_k+1	can	" "	" "
<s>	<s>	PRO	PRO t''	PRO	PRO	PRO	<s>	<s>
		D	D	D t'	D	D		
		N	N	N	N	N		
		V	V	V	V t	V		

Definition der Backward-Wahrscheinlichkeiten $\beta_{t'',t'}(k)$:

$$\beta_{t'',t'}(n+2) = \begin{cases} 1 & \text{falls } t'' = t' = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$$

$$\beta_{t'',t'}(k) = \sum_{t \in T} p(t|t'',t')\; p(w_{k+1}|t)\; \beta_{t',t}(k+1) \quad \text{für } 0 \le k \le n+1$$

Der Musterlösung-Code wurde implementiert nach der Formel
B_t'',t'(k-1) = Sum_t p(t|t'',t') p(w_k| t) B_t',t (k) für 0 < k <= n+2

Unterscheid zwischen beide Formeln -> Nächste Seite

```
def backward(words):
    words =  [''] + words + ['', '']  # Grenztokens hinzufügen

    # Initialisierung der Backward-Tabelle
    bwd_prob = [defaultdict(float) for _ in words]
    bwd_prob[-1][('<s>','<s>')] = 1.0
    for i in range(len(words)-1, 0, -1):
        tags = tagset if i > 2 else ['<s>']
        for tag1 in tags:
            for (tag2, tag3), prevp in bwd_prob[i].items():
                p = prevp * contextprob(tag1, tag2, tag3) * lexprob(words[i], tag3)
                bwd_prob[i-1][(tag1, tag2)] += p
    return bwd_prob
```

Definition der Backward-Wahrscheinlichkeiten $\beta_{t'',t'}(k)$:

$$\beta_{t'',t'}(n+2) = \begin{cases} 1 & \text{falls } t'' = t' = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$$

$$\beta_{t'',t'}(k) = \sum_{t \in T} p(t|t'',t')\; p(w_{k+1}|t)\; \beta_{t',t}(k+1) \quad \text{für } 0 \le k \le n+1$$

Bei dieser Formel, berechnen wir an Position k=n+1 rückwarts bis k=0 die Backward-Wk (für Position k).
Im Fall unseres Beispiels sind es Positionen 6,5,4,3,2,1,0
Z.B wenn k=6, berechnen wir alle Wk für diese Position (also Position 6)
(Für Position 7 wurde die Wk bei der Initialisierung schon berechnet)

$$\beta_{t'',t'}(k) = \sum_{t \in T} p(t|t'',t')\; p(w_{k+1}|t)\; \beta_{t',t}(k+1) \quad \text{für } 0 \le k \le n+1$$

$$\beta_{t'',t'}(n+2) = \begin{cases} 1 & \text{falls } t'' = t' = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$$

	0	1	2	3	4	5	6	7
“ “	“ “	I	can	can	a	can	“ “	“ “
<s>	<s>	PRO D N V	PRO D N V	PRO D N V	PRO D N V	PRO D N V	<s>	<s>

Definition der Backward-Wahrscheinlichkeiten $\beta_{t'',t'}(k)$:

$$\beta_{t'',t'}(n+2) = \begin{cases} 1 & \text{falls } t'' = t' = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$$

Umgeformte Formel

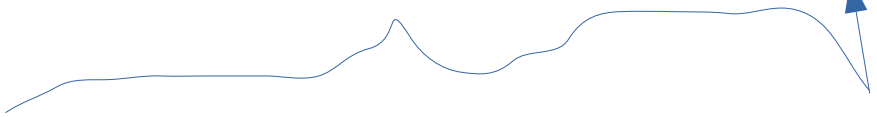
B_t'',t'(k-1) = Sum_t p(t|t'',t') p(w_k| t) B_t',t (k) für 0 < k <= n+2

Bei der umgeformten Formel, berechnen wir an Position k=n+2 rückwarts bis k=1 die Backward-Wk für Position **k-1 (und nicht k)**.
Im Fall unseres Beispiels sind k die Positionen 7,6,5,4,3,2,1.
Der Unterscheid zu der andere Formel ist, hier an Position k berechnen wir die Wk für Position k-1.
Deswegen iterieren wir von Position 7 bis 1 (und nicht 6 bis 0 wie bei der anderen Formel).
Z.B. An k= 7 wird die Backward-Wk für k=6 berechnet. (Backward-Wk für k=7 wurde vorher initializiert)
Das ist was der Code von der Musterlösung macht.

Umgeformte Formel

B_t'',t'(k-1) = Sum_t p(t|t'',t') p(w_k| t) B_t',t (k) für 0 < k <= n+2

$$\beta_{t'',t'}(n+2) = \begin{cases} 1 & \text{falls } t'' = t' = \langle s \rangle \\ 0 & \text{sonst} \end{cases}$$



	0	1	2	3	4	5	6	7
" "	" "	I	can	can	a	can	" "	" "
<s>	<s>	PRO D N V	PRO D N V	PRO D N V	PRO D N V	PRO D N V	<s>	<s>

Beide Formeln liefern das gleiche Ergebnis. Also, Position 0 bis n+2 bekommen die Wahrscheinlichkeiten für jede mögliche Tagpaar.

```
def backward(words):
    words =  [''] + words + ['', '']  # Grenztokens hinzufügen

    # Initialisierung der Backward-Tabelle
    bwd_prob = [defaultdict(float) for _ in words]
    bwd_prob[-1][('<s>','<s>')] = 1.0
    for i in range(len(words)-1, 0, -1):
        tags = tagset if i > 2 else ['<s>']
        for tag1 in tags:
            for (tag2, tag3), prevp in bwd_prob[i].items():
                p = prevp * contextprob(tag1, tag2, tag3) * lexprob(words[i], tag3)
                bwd_prob[i-1][(tag1, tag2)] += p
    return bwd_prob
```

Erklärung für diese Zeile

Der Grund für diese Zeile ist, wir wollen, dass alle Grenzpositionen nur <s> als der einzige mögliche Tag haben (So wie in der Darstellung), andere Positionen bekommen PRO,D,N,V aus tagset.

Bei der umgeforten Formel iterieren wir in dem ersten For-Loop über Position 7,6,5,..,1.
Die variable tags ist die Liste von möglichen Tags für tag1 (entspricht t'').

In dieser Zeile steht, für i > 2 ist tags gleich tagset (bestehend aus mehrerer Tags), sonst besteht tags nur aus <s>.

D.h. an i = 3,4,5,6,7 kommt tag1 aus mehreren möglichen Tags und an i=2,1 hat tag1 nur <s> als der einzige mögliche Tag.

Betrachten wir jetzt die Formel.
Als wir an Position i sind, (um die Wk für i-1 zu berechnen), definieren wir tag1,tag2,tag3. In der Darstellung sehen wir, dass tag3(t) der Tag an Position i ist. tag2 (t') ist der Tag an i-1, **und tag1(t'') ist der Tag an i-2.**

Für i=2, muss tag1 <s> sein, weil tag1 der Tag von Position 0 ist.
Für i=1, muss tag1 auch <s> sein, weil tag1 der Tag von Position -1.

Bei dem Forward-Code gilt ähnliche Erklärung.

Umgeformte Formel

$B\_t'',t'(k-1) = \text{Sum\_t}\ \ p(t|t'',t')\ p(w\_k|\ t)\ \ B\_t',t\ (k)\ \ \text{für}\ \ 0 < k <= n+2$

		i-1	i					
	0	1	2	3	4	5	6	7
" "	" "	I	can	can	a	can	" "	" "
<s>	<s>	PRO	PRO	PRO	PRO	PRO	<s>	<s>
	t''	D t'	D N t	D	D	D		
		N	N	N	N	N		
		V	V	V	V	V		

Beipiel Output (ohne tag PRO)

	0	1	2	3	4	5	6	7
“ “	“ “	I	can	can	a	can	“ “	“ “
<s>	<s>	D N V	D N V	D N V	D N V	D N V	<s>	<s>

```
position 0
('<s>', '<s>') 0.01
```

```
position 1
('<s>', 'N') 0.01
('<s>', 'D') 0.02
('<s>', 'V') 0.0
```

```
position 2
('N', 'N') 0.02
('N', 'D') 0.02
('N', 'V') 0.01
('D', 'N') 0.02
('D', 'D') 0.01
('D', 'V') 0.02
('V', 'N') 0.04
('V', 'D') 0.01
('V', 'V') 0.0
```

```
position 3
('N', 'N') 0.02
('N', 'D') 0.05
('N', 'V') 0.02
('D', 'N') 0.02
('D', 'D') 0.02
('D', 'V') 0.01
('V', 'N') 0.02
('V', 'D') 0.02
('V', 'V') 0.0
```

```
position 4
('N', 'N') 0.03
('N', 'D') 0.03
('N', 'V') 0.01
('D', 'N') 0.03
('D', 'D') 0.05
('D', 'V') 0.02
('V', 'N') 0.02
('V', 'D') 0.01
('V', 'V') 0.01
```

```
position 5
('N', 'N') 0.04
('N', 'D') 0.02
('N', 'V') 0.03
('D', 'N') 0.0
('D', 'D') 0.06
('D', 'V') 0.06
('V', 'N') 0.02
('V', 'D') 0.02
('V', 'V') 0.02
```

```
position 6
('N', '<s>') 0.06
('D', '<s>') 0.27
('V', '<s>') 0.12
```

```
position 7
('<s>', '<s>') 1.0
```

Wie wird die Backward-Wk bei der ungeänderten Formel berechnet?
Beipiel: Wir wollen hier B(N,<s>, 6) berechnen.

	0	1	2	3	4	5	k	k+1
“ “	“ “	I	can	can	a	can	6 “ “	7 “ “
<s>	<s>	D N V	D N V	D N V	D N V	D N V t”	<s> t’	<s> t

```
position 6
('N', '<s>') 0.06
('D', '<s>') 0.27
('V', '<s>') 0.12
```

```
position 7
('<s>', '<s>') 1.0
```

$$\beta_{t'',t'}(k) \quad = \quad \sum_{t \in T} p(t|t'',t')\; p(w_{k+1}|t)\; \beta_{t',t}(k+1) \qquad \text{für } 0 \leq k \leq n+1$$

Als wir an Position k ist, berechnen wie die Backward-Wk für Position k

Man kann einen Code nach dieser Formel auch schreiben

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

```python
def backward(words):
    words =  [''] + words + ['', '']  # Grenztokens hinzufügen

    # Initialisierung der Backward-Tabelle
    bwd_prob = [defaultdict(float) for _ in words]
    bwd_prob[-1][('<s>','<s>')] = 1.0
    for i in range(len(words)-2, -1, -1):
        tags = tagset if i > 1 else ['<s>']
        for tag1 in tags:
            for (tag2, tag3), prevp in bwd_prob[i+1].items():
                p = prevp * contextprob(tag1, tag2, tag3) * lexprob(words[i+1], tag3)
                bwd_prob[i][(tag1, tag2)] += p
    return bwd_prob
```

Bigramm

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

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

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

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

Link zum Code:
https://colab.research.google.com/drive/1NH7FNCaMsRkHGLBv9QvZn0Q0m-dclcil?usp=sharing