1. Exercise Sheet

Advanced Automatic Speech Recognition

RWTH Aachen — Patrick von Platen

Prof. Dr.Ing H. Ney

Exercise 1

a)

We know that the following equation holds

$$\sum_{N=1}^{\infty} p(N) = 1 \tag{1}$$

Knowing that we have a bigram language model, we can derive:

$$\begin{aligned}
\sum_{N=1}^{\infty} p(N) \sum_{w_1^N} p(w_1^N) &= \sum_{N=1}^{\infty} p(N) \sum_{w_1^N} \prod_{i=1}^{N} p(w_i|w_{i-1}) \\
\sum_{N=1}^{\infty} p(N) \sum_{w_1^N} p(w_1^N) &= \sum_{N=1}^{\infty} p(N) \sum_{w_1^{N-1}} \prod_{i=1}^{N-1} p(w_i|w_{i-1}) \sum_{w_N} p(w_N|w_{N-1}) \\
\sum_{N=1}^{\infty} p(N) \sum_{w_1^N} p(w_1^N) &= \sum_{N=1}^{\infty} p(N) \sum_{w_1^{N-1}} \prod_{i=1}^{N-1} p(w_i|w_{i-1}) 1 \\
(\ldots \textit{ recursively}) \quad &= \sum_{N=1}^{\infty} p(N) \prod_{i=1}^{N} 1 \\
&= \sum_{N=1}^{\infty} p(N) \\
(\textit{using equation 1}) \quad &= 1
\end{aligned}$$

b)

i)

The sentence end token on language model allows for language model normalization on word level independent of the length of the sentence. Without the sentence end token the following must always be true:

$$\prod_{i=1}^{N} w_i = (\prod_{i=1}^{N-1} w_i) w_N \leq \prod_{i=1}^{N-1} w_i$$

The above equation logically holds, but having a corpus with a finite amount of sentences, it might very well be the case, that a sentence containing a subsentence is more likely to occur as a full sentence than the subsentence as a full sentence.

Therefore, the sentence end model makes it possible to include sentence ends in the model and in doing so also ensures language model normalization on word level

$$\sum_{w \in \mathbb{V} \cup \{\$\}} p(w|v) = 1, \forall v \in \mathbb{V}$$

ii)

$$\begin{aligned}
\sum_{N=1}^{\infty} \sum_{\substack{w_1^N : w_N = \$ \\ w_n \in \mathbb{V}, \forall n = 1, \ldots, N-1}} p(w_1^N) &= \sum_{N=1}^{\infty} \sum_{\substack{w_1^N : w_N = \$ \\ w_n \in \mathbb{V}, \forall n = 1, \ldots, N-1}} \prod_{i=1}^{N} p(w_i|w_{i-1}) \\
&= \sum_{N=1}^{\infty} \sum_{w_1^{N-1}, w_n \in \mathbb{V}} \prod_{i=1}^{N-1} (p(w_i|w_{i-1})) p(\$|w_{i-1}) \\
&= \sum_{N=1}^{\infty} \sum_{w_1^{N-1}, w_n \in \mathbb{V}} \prod_{i=1}^{N-1} (p(w_i|w_{i-1})) p(\$) \\
&= \sum_{N=1}^{\infty} \sum_{w_1^{N-2}, w_n \in \mathbb{V}} \prod_{i=1}^{N-2} (p(w_i|w_{i-1}) \sum_{w_{N-1}} p(w_{N-1}|w_{N-2})) p(\$) \\
&= \sum_{N=1}^{\infty} \sum_{w_1^{N-2}, w_n \in \mathbb{V}} \prod_{i=1}^{N-2} (p(w_i|w_{i-1})(1 - p(\$)) p(\$) \\
(\ldots \textit{ recursively}) &= \sum_{N=1}^{\infty} (1 - p(\$))^{N-1} p(\$) \\
&= \sum_{N=1}^{\infty} (1 - p(\$))^{N-1} p(\$) \\
(\textit{using geometric series}) &= \frac{1}{1 - (1 - p(\$))} p(\$) \\
&= 1
\end{aligned}$$

iii)

We can imply from ii) that

$$p(N) = \sum_{w_1^N : w_N = \$, w_n \in \mathbb{V}, \forall n = 1, \ldots, N-1} \prod_{i=1}^{N} p(w_i|w_{i-1}) = (1 - p(\$))^{N-1} p(\$)$$

Exercise 2

a)

Assuming that the language model uses a bigram and that the model $p(w_n, t_n|t_{n-1}, w_{n-1}, x_1^T)$ is given, we define:

$$Q(t,w) = \max_{\substack{w_1^n, t_1^n, n \\ w_n=w, t_n=t}} \prod_{k=1}^{n} p(w_k, t_k|t_{k-1}, w_{k-1}, x_1^T)$$

b)

Let's derive our recursive formula:

$$\begin{aligned} Q(t,w) &= \max_{\substack{w_1^n, t_1^n, n \\ w_n=w, t_n=t}} \prod_{k=1}^{n} p(w_k, t_k|t_{k-1}, w_{k-1}, x_1^T) \\ &= \max_{l\in[1,t-1],v} \max_{\substack{w_1^n, t_1^n, n \\ w_n=w, t_n=t \\ n-1, w_{n-1}=v, t_{n-1}=l}} \prod_{k=1}^{n-1} p(w_k, t_k|t_{k-1}, w_{k-1}, x_1^t) p(w,t|l,v,x_1^l) \\ &= \max_{l\in[1,t-1],v} Q(l,v) p(w,t|l,v,x_1^l) \end{aligned}$$

c)

We need to store both the best predecessor word for every word V and its word boundary L

$$(L,V)(t,w) = \text{argmax}_{l\in[1,t-1],v} Q(l,v) p(w,t|l,v,x_1^l)$$

d)

First, let's define some variables:

- T = time length of signal
- W = size of vocabulary
- $\overline{L}$ = average time length of word

The time complexity of dynamic programming is for a bigram $T \times T \times W \times W$,since we try out the language model recombination at every time t and word w for all wordsat every earlier word boundary time $l \leq t-1$. Since one T is always less than actual T it will be a little smaller than the above mentioned.

The memory complexity of dynamic programming for a bigram is $2 \times T \times W$ since we need to store both t and w value for L and V