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:

$$\begin{aligned}
\sum_{N=1}^{\infty} p(N) \sum_{w_1^N} p(w_1^N) &= \sum_{N=1}^{\infty} p(N) \sum_{w_1^N} \prod_{i=1}^{N} p(w_i|w_{i-1}) \\
(\textit{sum over all words in vocab} \quad &= \sum_{N=1}^{\infty} p(N) \prod_{i=1}^{N} \sum_{w_i} p(w_i|w_{i-1}) \\
&= \sum_{N=1}^{\infty} p(N) \prod_{i=1}^{N} 1 \\
&= \sum_{N=1}^{\infty} p(N) \\
(\textit{using equation } 1) \quad &= 1
\end{aligned}$$

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)

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

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:

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

b)

Let's derive our recursive formula:

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

c)

Since every time frame can represent a word boundary, we need to store the best word ending at this time frame. Additionally, we need to know, the best previous word boundary for every time frame t to be able to connect the best words to its best predecessor words. Thus we have the best word ending at time t:

$$W(t) = \text{argmax}_w Q(t, w)$$

and the best previous word boundary

$$L(t) = L(t, W(t))$$

with

$$L(t, w) = \text{argmax}_l Q(l, W(l)) p(t, w | l, W(l), x_1^t)$$

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 $T \times T \times W \times T$,since we try out the language model recombination at every time t and word w for all wordsat every earlier word boundary time $l \leq t-1$. The memory complexity of dynamic programming for a bigram is $\overline{L} \times W + 2 \times T$, since first we need to store on average for $\overline{L}$ time instances the scores of every word and then we also need to store the best words and previous word boundaries for every time instance of our traceback arrays.