This is an optional reading about perplexity computation to make sure you remember the material of the videos and you are in a good shape for the quiz of this week.

Perplexity is a popular quality measure of language models. We can calculate it using the formula:

$$\mathcal{P} = p(w_{test})^{-\frac{1}{N}}, \text{ where } p(w_{test}) = \prod_{i=1}^{N+1} p(w_i | w_{i-1}^{i-n+1})$$

Recall that all words of the corpus are concatenated and indexed in the range from 1 to N. So N here is ***the length of the test corpus***. Also recall that the tokens out of the range are ***fake start/end tokens*** to make the model correct.

Check yourself: how many start and end tokens do we have in a trigram model?

Now, if just one probability in the formula above is equal to zero, the whole probability of the test corpus is zero as well, so the perplexity is infinite. To avoid this problem, we can use different methods of smoothing. One of them is Laplacian smoothing (add-1 smoothing), which estimates probabilities with the formula:

$$p(w_i | w^{i-1}_{i-n+1}) = \frac{c(w^i_{i-n+1}) + 1}{c(w^{i-1}_{i-n+1}) + V}$$

$$p(w_i | w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i-1}) + V}{c(w_{i-n+1}^{i-1}) + 1}$$

Note, that V here is the number of possible continuations of the sequence w^{i-1}_{i-n+1}, so V is ***the number of unique unigrams in the train corpus plus 1***. Do you see why? Well, we include the fake end token to this number, because the model tries to predict it each time, just us as any other word. And we do not include the start tokens, because they serve only as a prefix for the first probabilities.

Now, let's review the following task together.

Task:

Apply add-one smoothing to the trigram language model trained on the sentence:

"This is the cat that killed the rat that ate the malt that lay in the house that Jack built."

Find the perplexity of this smoothed model on the test sentence:

"This is the house that Jack built."

Solution:

We have n=3, so we will add two start tokens <s1>, <s2> and one end token <end>.

Note, that we add ***(n-1) start tokens*****,** since the start tokens are needed to condition the probability of the first word on them. The role of the end token is different and we always add ***just one end token****.* It's needed to be able to finish the sentence in the generative process at some point.

So, what we have is:

train: *<s1> <s2> This is the cat that killed the rat that ate the malt that lay in the house that Jack built <end>*

test: *<s1> <s2> This is the house that Jack built <end>*

Number of unique unigrams in train is 14, so V = 14 + 1 = 15.

Number of words in the test sentence is 7, so N = 7.

$\mathcal{P}=p(w_{test})^{-\frac{1}{N}}$, where $p(w_{test})=\prod_{i=1}^{8}p(w_i|w_{i-2}w_{i-1})=\prod_{i=1}^{8}\frac{c(w_{i-2}w_{i-1}w_i)+1}{c(w_{i-2}w_{i-1})+15}$

All right, now we need to compute 8 conditional probabilities. We can do it straightforwardly or notice a few things to make our life easier.

First, note that all bigrams from the test sentence occur in the train sentence ***exactly once*****,** which means we have (1 + 15) in all denominators.

Also note, that "is the house" is the only trigram from the test sentence that is not present in the train sentence. The corresponding probability is p(house | is the) = (0 + 1) / (1 + 15) = 0.0625.

All other trigrams from the test sentence occur in the train sentence exactly once. So their conditional probabilities will be equal to (1 + 1) / (1 + 15) = 0.125.

In this way, perplexity is (0.0625 * 0.125 ^ 7) ^ (-1/7) = 11.89.