Joshua Wang
10/2/22
CS381 NLP

Note: Most calculations were made trying to exclude the <s> tag unless stated otherwise. Also, since calculations were done using unsimplified results in python, certain results may be slightly different due to decimals.

PART I

3.4 We are given the following corpus, modified from the one in the chapter:

```
<s> I am Sam </s>
<s> Sam I am </s>
<s> I am Sam </s>
<s> I do not like green eggs and Sam </s>
```

Using a bigram language model with add-one smoothing, what is P(Sam | am)? Include <s> and </s> in your counts just like any other token.

P(Sam|am) = am Sam / am = 2/3
V = 10
With add one smoothing = (2+1 / 3+10) = 3/13.

PART II Questions:

1. The amount of unique words in the training corpus, with the <s> tag excluded, is 39,502.
2. The amount of word tokens in the training corpus, with the <s> tag excluded, is 2,221,290.
3. The percentage of word types in the test corpus non matching with training data, and excluding the <s> tag is ~3.92%.
   The percentage of word tokens in the test corpus non matching with training data, and excluding the <s> tag is ~1.81%.
4. The percentage of bigram types in the test corpus non matching with training data, and excluding the <s> tag is ~37.05%.
   The percentage of bigram tokens in the test corpus non matching with training data, and excluding the <s> tag is ~22.32%.
5. Calculating the log probability of the sentence “I look forward to hearing your reply.” under the three models:
   Unigram Model: To find log probability of our sentence using the unigram model, we will need the count of each sentence’s individual word tokens in our training data, and the overall number of word tokens. We find the probability of each word as that word’s training token counts over total tokens in training data. Then by applying log to the result, we get a log probability. Then we can add each log probability to find the overall

probability of the sentence. A formula for our sentence may look like this: log(c(i)/c(total tokens)) + log(c(look)/c(total tokens)) + log(c(forward)/c(total tokens)) + log(c(to)/c(total tokens)) + log(c(hearing)/c(total tokens)) + log(c(your)/c(total tokens)) + log(c(reply)/c(total tokens)) + log(c(.)/c(total tokens)). Using these calculations and including the </s> tag in it, the log probability of this sentence is about -89.82.

Bigram Model: To find log probability of our sentence using the bigram model, we will need our sentence and training data sorted into bigrams. The parameters we will need are the count of our sentence's bigrams in the training data, which will be our word probability's numerator and the total count of each word, which will be used for each word probability's denominator. Since we are dealing with log probability, we will also apply log to each individual word probability and add them to get the total probability of our sentence. A formula for our sentence may look like this: log(c(<s> i)/c(<s>)) + log(c(i look)/c(i)) ... + log(c(. </s>)/c(.)). However, because the bigram 'hearing your' does not appear in our training data, it has a 0% probability and makes it hard to viably calculate the probability of the whole sentence.

Bigram Add One Smoothing Model: Formula and parameters are nearly identical to regular bigram model except this time 1 is added to the numerator and training vocab size is added to the denominator of probability calculations. Now unseen bigrams like "hearing your" do not result in a 0 or undefined probability, with total log probability being about -97.40.

6. Since the log probability of our sentence can't be computed with a bigram model, perplexity will only be calculated for a unigram and bigram add one smoothing model from now on.

   Unigram: The formula for perplexity is 2^-L, with L being the per word token probability, which is just our total sentence log probabilities divided by the number of sentence tokens. Our unigram total log probability is about -89.82, and the total number of word tokens in our sentence excluding <s> is 9. L = -89.82/9 = -9.98. P = 2^-(-9.98) = ~1009.81 Therefore our sentence's unigram perplexity is about 1,009.81.

   Bigram Add One: Our bigram smoothed total log probability is about -97.40 and there are 9 word tokens. L = -97.40/9 = -10.82. P = 2^-(-10.82) = ~1,810.72. Therefore our sentence's unigram perplexity is about 1,810.72.

7. The first step to finding our test corpus's perplexity will be finding its total log probability.

   Unigram: Just like when we computed the unigram log probability of our test sentence, we calculate total probability as the summation of the log base 2 applied count of each training word token (except <s>) over the count of total training word tokens. The total test corpus log probability is about -27,900.07, and with 2,769 test word tokens, our per token probability, L, is about -10.08. We calculate perplexity as 2^-L, which is about 1,079.29 for our unigram test corpus.

Bigram Add One: For bigram smoothed perplexity, we will also need to calculate total log probability of the test corpus. We can calculate total probability as the summation of the log base 2 applied count of each bigram + 1, over the count of the previous token(first half of our bigram) + training vocab size. The total test corpus log probability is about -31,041.26, and with 2,769 test word tokens, our per token probability, L, is about -11.21. We calculate perplexity as 2^-L, which is about 2,369.36 for our bigram smoothed test corpus.

Result Discussion: While normally one might expect a bigram language model to be a better language model than a unigram one, the unigram model is statistically better with a lower perplexity score of 1,079.29 compared to the bigram model's 2,369.36 score. Looking at the results of q3 and q4, this is to be expected as there are far more matching individual word tokens and types between the test and training models compared to very rarely matching bigram tokens and types. I believe the combinations of already seen words in training data suddenly becoming "unseen" as a bigram may play a big role in this perplexity difference. Perhaps if training data was significantly greater in size, we might see different results.