NLP - Exercise 2

Student ID: 211406343

Part 1:

In this part, I implemented a method in the NGramModel class to calculate the probability for a sentence in a trigram model using linear interpolation. In this method, I used three parameters ($\lambda_1, \lambda_2, \lambda_3$) as described in the lecture of N-Grams models:

1. λ_1 is multiplied by the probability of having the sequence of the last three words.
2. λ_2 is multiplied by the probability of having the sequence the last two words.
3. λ_3 is multiplied by the probability of having the last word.

I set λ_1 to be 0.8, λ_2 to 0.15 and λ_3 to be 0.05 since their sum must be 1.

Why these values?

If the sequence of the last three words in the sentence appeared several times in the corpus, then it probably has a meaning, and we should give its appearance a greater weight than the appearance of a single word or two consecutive words.

For the given sentences in the exercise, to get the maximal probability in the trigram part, we need to set λ_1 value to 1 which makes since according to what we have discussed in the previous paragraph. However, setting the value of λ_1 to 1 means we do not consider neither the part of probability of the last word by itself nor the probability of the last couple of words, thus we will not have an interpolation. As a result, once we get a sentence that has a sequence of 3 words which has never appeared in the corpus before, the probability of the whole sentence will be equal to 0.

Therefore, we set the value of λ_1 to be great but we shall keep it away from 1.

In addition, I implemented another two methods in the same class (NGramModel), one is for calculating the probability of a sentence in Unigram model using Laplace smoothing, while the other is for calculating the probability of a sentence in Bigram model using the same algorithm for smoothing.

Note that:

1. The class NGramModel had a class attribute of the type of Corpus which helped me iterate over the sentences in the corpus and process them.

2. While calculating the probability in each method from above, I had to sum the log for the probabilities in base ***e*** instead of multiplying them because their values are so close to zero. A multiplication might make the result equal to zero which is incorrect.

Part 2:

1. Yes, but very few. Some parts of the Trigram model sentences make sense. However, the Unigram had terrible results, and the Bigram had bad results but they still better than the Unigram's ones.

2. As we can see, the larger N is in N-Gram model, the better the results are in generating sentences. That is because Trigrams – for example - searched for a longer sequence in each step. Therefore, its sentences had more meaningful phrases from the corpus and well-related words. Even on the grammar side, the sentences were mostly correct.

3. The sentences generated by 6-grams models will be meaningful and will be more well-connected than Trigrams' ones which just had several inner subsentences in each sentence. But the 6-grams model will not have as many sentences to generate as the Trigrams have. That is because 6-grams model searches for words following the last 5 words the current sentence has which are very few compared to the ones following the last 2 words. Thus, we might stop at a fair value for N for a trade-off between having meaningful sentences and having a good number of random sentences to generate. In addition, we might not use a very high value for N just for a little improved results since the larger N is, the worse the time complexity gets in a dramatic way.

For this part I implemented a method in Corpus class which returns a random sentence length from the corpus. Then used this length for generating sentences for each model (Unigram, Bigram and Trigram).