Natural Language Processing
Lab 2

This lab sheet is to practice the concepts taught this week far with a focus on n-gram language models.

1. Write out the equation for trigram probability estimation (modifying Eq. 3.11 from SLP Chapter 3). Now write out all the non-zero trigram probabilities for the I am Sam corpus in Chapter 3 on page 4.

Ans: $P(w_n|w_{n1}, w_{n2}) = \frac{C(w_{n2},w_{n1},w_n)}{C(w_{n2},w_{n1})}$

2. Write a program to compute unsmoothed n-grams. Use the Dr. Seuss corpus to test this.

Ans:

```
def tokenize(text):
    """ Tokenize the text into words. """
    return text.split()

def generate_ngrams(tokens, n):
    """ Generate n-grams from tokens. """
    ngrams = []
    for i in range(len(tokens)-n+1):
        ngram = ' '.join(tokens[i:i+n])
        ngrams.append(ngram)
    return ngrams

def count_ngrams(ngrams):
    """ Count the frequency of each n-gram. """
    ngram_counts = {}
    for ngram in ngrams:
        ngram_counts[ngram] = ngram_counts.get(ngram, 0) + 1
    return ngram_counts
```

```python
def calculate_probabilities(ngrams, n_minus_one_grams):
    """ Calculate the probability of each n-gram. """
    probabilities = {}
    for ngram in ngrams:
        n_minus_one_gram = ' '.join(ngram.split()[:-1])
        probabilities[ngram] = ngrams[ngram] / n_minus_one_grams[n_minus_one_gram]
    return probabilities

# Example usage
text = """<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>"""
tokens = tokenize(text)
n = 2  # For bigrams

# Generate and count n-grams and (n-1)-grams
ngrams = generate_ngrams(tokens, n)
ngram_counts = count_ngrams(ngrams)
n_minus_one_grams = generate_ngrams(tokens, n-1)
n_minus_one_counts = count_ngrams(n_minus_one_grams)

# Calculate probabilities
ngram_probabilities = calculate_probabilities(ngram_counts, n_minus_one_counts)

for ngram, probability in ngram_probabilities.items():
    print(f"'{ngram}': {probability:.4f}")
```

3. Run your n-gram program on two different small corpora of your choice (you might use email text or newsgroups). Now compare the statistics of the two corpora. What are the differences in the most common unigrams between the two? How about interesting differences in bigrams?

Ans: A good approach to this problem would be to sort the N-grams foreach corpus by their probabilities, and then examine the first 100-200 for each corpus. Both lists should contain the common functionwords, e.g., the,of,to, etc near the top. The content words are proba-bly where the more interesting differences are – it should be possibleto see some topic differences between the corpora from these.

4. Write a definition of perplexity in language modeling.

Ans: Perplexity is a key metric in language modelling, often used to evaluate the performance of a language model. It's a measure of how well a probability model predicts a sample. It is defined in terms of the probability assigned to a sequence of words by a language model. It's a measure of the model's ability to predict or generate a given text. A lower perplexity score indicates a better model. It means that the model assigns higher probabilities, on average, to the sequences it encounters in the test data. This implies that the model is more effective at predicting or generating text that resembles the test data. Perplexity is also related to the concept of entropy, a key idea in information theory. In a sense, it measures the amount of "surprise" or "uncertainty" a model has when predicting new data.

In summary, perplexity is a measure of a language model's effectiveness, gauging how likely the model is to predict a given sequence of words. It's widely used for comparing different models or tuning the parameters of a single model. A model that can predict text more accurately (i.e., with fewer "surprises") will have a lower perplexity.

5. Add an option to your program to compute the perplexity of a test set.

Ans:

```
        # probably a number of ways to do this, but it should capture the followi
        perplexity = probability_sequence ** (-1 / length_sequence)
```

6. You are given a training set of 100 numbers that consists of 91 zeros and 1 each of the other digits 1-9. Now we see the following test set: 0 0 0 0 0 0 3 0 0 0 0. What is the unigram perplexity?

Ans:

```
# The following outlines the methodical approach to take to think about this probl
p_zero = 91 / 100
p_digit = 1 / 100  # Same for digits 1-9

# Test set: 9 zeros and 1 three
probability_sequence = (p_zero ** 9) * p_digit
length_sequence = 10  # Length of the test set
```

```
# Unigram perplexity
perplexity = probability_sequence ** (-1 / length_sequence)
perplexity
```