CHP 2: N-Gram Language Model

INTRODUCTION TO N-GRAM

An N-Gram is a sequence of NNN items (words or characters) from a text. The value of NNN determines the type of N-Gram:

- **Unigram (1-Gram)**: Single item.
  - **Example**:
    - Text: "The cat sat"
    - Unigrams: ['The', 'cat', 'sat']
- **Bigram (2-Gram)**: Sequence of two items.
  - **Example**:
    - Text: "The cat sat"
    - Bigrams: ['The cat', 'cat sat']
- **Trigram (3-Gram)**: Sequence of three items.
  - **Example**:
    - Text: "The cat sat"
    - Trigrams: ['The cat sat']
- **4-Gram**: Sequence of four items.
  - **Example**:
    - Text: "The cat sat on the mat"
    - 4-Grams: ['The cat sat on', 'cat sat on the', 'sat on the mat']

N-Gram probability estimation and perplexity

Example Sentence: "Cats are cute"

1. Unigram Probability

Unigram Probability is the probability of a single word occurring in a corpus.

For the sentence **"Cats are cute"**:

- **Total Count of Words**: 3 (i.e., "Cats", "are", "cute")
- **Count of Each Word**:
  - Count(Cats) = 1
  - Count(are) = 1
  - Count(cute) = 1

Unigram Probabilities:

- $P(\text{Cats}) = \frac{\text{Count(Cats)}}{\text{Total Count}} = \frac{1}{3} \approx 0.33$
- $P(\text{are}) = \frac{\text{Count(are)}}{\text{Total Count}} = \frac{1}{3} \approx 0.33$
- $P(\text{cute}) = \frac{\text{Count(cute)}}{\text{Total Count}} = \frac{1}{3} \approx 0.33$

2. Bigram Probability

Bigram Probability is the probability of a word occurring given the previous word.

For the sentence **"Cats are cute"**:

- **Count of Bigram Pairs**:
  - Count(Cats are) = 1
  - Count(are cute) = 1
- **Count of Each Word**:
  - Count(Cats) = 1
  - Count(are) = 1

Bigram Probabilities:

- $P(\text{are}|\text{Cats}) = \frac{\text{Count(Cats are)}}{\text{Count(Cats)}} = \frac{1}{1} = 1.0$
- $P(\text{cute}|\text{are}) = \frac{\text{Count(are cute)}}{\text{Count(are)}} = \frac{1}{1} = 1.0$

3. Trigram Probability

Trigram Probability is the probability of a word occurring given the two preceding words.

For the sentence "**Cats are cute**":

- **Count of Trigram Pairs**:
  - Count(Cats are cute) = 1
  - Count(Cats are) = 1

Trigram Probability:

- $P(\text{cute}|\text{Cats are}) = \frac{\text{Count(Cats are cute)}}{\text{Count(Cats are)}} = \frac{1}{1} = 1.0$

Summary

- **Unigram Probability** measures the likelihood of a single word occurring in the dataset.
- **Bigram Probability** measures the likelihood of a word occurring given the previous word.
- **Trigram Probability** measures the likelihood of a word occurring given the two preceding words.

Let's illustrate this with a simple table for the given example:

Word Sequence	Count	Total Count	Probability
Cats	1	3	0.33
are	1	3	0.33
cute	1	3	0.33
Cats are	1	1	1.0
are cute	1	1	1.0
Cats are cute	1	1	1.0

Step-by-Step Calculation of Perplexity

Step-by-Step Calculation of Perplexity

Sentence: "Cats are cute"

1. **Calculate N-gram Probabilities**:

   From the previous example, we computed:

   - $P(\text{are}|\text{Cats}) = 1.0$
   - $P(\text{cute}|\text{are}) = 1.0$

2. **Calculate the Probability of the Sentence**:

   For a sentence w_1, w_2, w_3, the probability is calculated using bigrams and trigrams:

   - **Bigram Probability**:
     - $P(\text{are}|\text{Cats}) = 1.0$
     - $P(\text{cute}|\text{are}) = 1.0$

   The probability of the entire sentence is the product of these probabilities:

   $$P(\text{Cats are cute}) = P(\text{are}|\text{Cats}) \times P(\text{cute}|\text{are})$$

   $$P(\text{Cats are cute}) = 1.0 \times 1.0 = 1.0$$

3. **Calculate Perplexity**:

   Perplexity PP is given by:

   $$\text{PP} = \left(\frac{1}{P(\text{Sentence})} \right)^{\frac{1}{N}}$$

   Where N is the number of words in the sentence.

   - **Number of Words (N)**: 3 (Cats, are, cute)

   $$\text{PP} = \left(\frac{1}{1.0} \right)^{\frac{1}{3}} = 1.0$$

Explanation:

- **Probability Calculation**: The probability of "Cats are cute" is 1.0, meaning the model perfectly predicts the sentence.

- **Perplexity Calculation**: Since the probability is 1.0, perplexity is also 1.0.

Interpretation:

A perplexity of 1.0 indicates that the model is very confident about the sentence and predicts it perfectly. In practical scenarios, a lower perplexity indicates better model performance, while a higher perplexity suggests the model is less confident or has poor predictive power.

```
Low Perplexity → Good Model Prediction
High Perplexity → Poor Model Prediction
```

Smoothing technique(Laplace/good Turing/Kneser-Ney/Interpolation

Laplace Smoothing, also known as **Add-One Smoothing**, is a technique used to handle the problem of zero probabilities in probabilistic models, particularly in language models. It ensures that no probability is zero, which is important when dealing with unseen words or word combinations.

What is Laplace Smoothing?

Laplace Smoothing adjusts the counts of n-grams by adding a small constant (usually 1) to all possible counts. This adjustment ensures that even previously unseen n-grams have a non-zero probability.

How Laplace Smoothing Works

For a given n-gram model, Laplace Smoothing modifies the probability calculation as follows:

1. **Count of N-grams**:
    - For each observed n-gram, add 1 to its count.
    - For each unobserved n-gram, assume a count of 1.
2. **Probability Calculation**:
    - Calculate the probability using the smoothed counts.

Example: Bigram Model

Text Corpus: "The cat sat on the mat."

Vocabulary: {The, cat, sat, on, mat}

1. Without Smoothing

Count of Bigrams:

- "The cat": 1
- "cat sat": 1
- "sat on": 1
- "on the": 1
- "the mat": 1

Total Count of Bigrams: 5

Probability Calculation:

- For "cat" given "The":

$$P(\text{cat} \mid \text{The}) = \frac{\text{Count}(\text{The cat})}{\text{Count}(\text{The})} = \frac{1}{1} = 1.0$$

- For an unseen bigram like "cat the":

$$P(\text{the} \mid \text{cat}) = \frac{\text{Count}(\text{cat the})}{\text{Count}(\text{cat})} = \frac{0}{1} = 0.0$$

Problem: Unseen bigrams have a probability of zero.

2. With Laplace Smoothing

Adjust Counts:

- Add 1 to each observed bigram count.
- Assume a count of 1 for unseen bigrams.

Vocabulary Size V=5V = 5V=5 (The, cat, sat, on, mat)

Smoothed Counts:

- "The cat": 1 + 1 = 2
- "cat sat": 1 + 1 = 2
- "sat on": 1 + 1 = 2
- "on the": 1 + 1 = 2
- "the mat": 1 + 1 = 2

- Unseen bigrams like "cat the": 0 + 1 = 1

Total Count of Bigrams (with smoothing): 5 (original) + 5 (for unseen bigrams) = 10

Smoothed Probability Calculation:

- For "cat" given "The":

$$P(\text{cat} \mid \text{The}) = \frac{\text{Count}(\text{The cat}) + 1}{\text{Count}(\text{The}) + V} = \frac{1+1}{1+5} = \frac{2}{6} \approx 0.33$$

- For an unseen bigram like "cat the":

$$P(\text{the} \mid \text{cat}) = \frac{\text{Count}(\text{cat the}) + 1}{\text{Count}(\text{cat}) + V} = \frac{0+1}{1+5} = \frac{1}{6} \approx 0.17$$

Summary

- **Laplace Smoothing** adjusts the counts by adding 1 to each observed bigram and assuming a count of 1 for unseen bigrams.
- **Probability Calculation** is updated to reflect these adjustments, ensuring that no bigram has a probability of zero.

Turing Smoothing (Good-Turing Smoothing)

Text Corpus: "The cat sat on the mat."

Vocabulary: {The, cat, sat, on, mat}

Concept: Good-Turing smoothing adjusts probabilities based on the frequency of counts. It estimates the probability of unseen bigrams by redistributing the probability mass of observed bigrams.

Steps:

1. **Count Frequencies**:
    - Count of Bigrams (as before): "The cat": 1, "cat sat": 1, "sat on": 1, "on the": 1, "the mat": 1
    - Bigram counts are all 1.

2. **Estimate Probabilities**:
   - Let N_1 be the count of bigrams that appear exactly once. Here, $N_1 = 5$.
   - Let N_2 be the count of bigrams that appear exactly twice. Here, $N_2 = 0$.
   - Estimate C^* (the smoothed count for a bigram that is observed c times):

$$C^* = \frac{(c+1) \cdot N_{c+1}}{N_c}$$

   For $c = 1$:

$$C^* = \frac{(1+1) \cdot N_2}{N_1} = \frac{2 \cdot 0}{5} = 0$$

   Here, since $N_2 = 0$, we use an alternative calculation for unseen bigrams.

3. **Calculate Probabilities**:
   - For "cat" given "The":

$$P(\text{cat} \mid \text{The}) = \frac{\text{Count}(\text{The cat})^*}{\text{Count}(\text{The})}$$

   With smoothing:

$$P(\text{cat} \mid \text{The}) = \frac{2}{5}$$

   - For unseen bigrams like "cat the": Use the adjusted probability for unseen bigrams (typically a small constant or redistributed probability).

```
Adjusts based on frequency counts of N-Grams
```

Kneser-Ney Smoothing

Concept: Kneser-Ney Smoothing builds on Good-Turing by incorporating context from lower-order models. It redistributes probability mass to unseen bigrams based on the likelihood of the preceding context.

Steps:

1. **Calculate Continuation Probability**:
    - Calculate the probability of the bigram in the context of lower-order (unigram) models.

2. Assign Probabilities:

- For observed bigrams, use:

$$P(\text{cat} \mid \text{The}) = \frac{\max(\text{Count}(\text{The cat}) - d, 0)}{\text{Count}(\text{The})} + \lambda(\text{The}) \cdot P_{\text{cont}}(\text{cat})$$

Where d is a discount factor, and λ is a normalization factor.

- For unseen bigrams:

$$P(\text{cat} \mid \text{The}) = \lambda(\text{The}) \cdot P_{\text{cont}}(\text{cat})$$

```
Discounted counts + Continuation probability
```

Interpolation Smoothing

Concept: Interpolation smoothing combines multiple levels of n-gram models (e.g., unigram, bigram) to estimate probabilities, allowing blending of different models.

Steps:

1. **Calculate Weighted Probabilities**:
   - Use weights for different models (unigram, bigram):
     $$P(\text{cat} \mid \text{The}) = \alpha \cdot P_{\text{bigram}}(\text{cat} \mid \text{The}) + \beta \cdot P_{\text{unigram}}(\text{cat})$$
     Where α and β are weights that sum to 1.
2. **Assign Probabilities**:
   - For observed bigrams:
     $$P(\text{cat} \mid \text{The}) = 0.7 \cdot \frac{2}{5} + 0.3 \cdot \frac{1}{5} = 0.7 \cdot 0.4 + 0.3 \cdot 0.2 = 0.28 + 0.06 = 0.34$$
   - For unseen bigrams:
     $$P(\text{cat} \mid \text{The}) = 0.3 \cdot \frac{1}{5} = 0.3 \cdot 0.2 = 0.06$$

```
Weighted average of unigram, bigram, trigram probabilities
```

Summary

- **Turing Smoothing** adjusts for unseen bigrams by redistributing the probability of observed bigrams.
- **Kneser-Ney Smoothing** refines this by leveraging context from lower-order models.
- **Interpolation Smoothing** combines multiple models to estimate probabilities, blending unigram and bigram probabilities.