🔹 Q12. Apply Laplace Smoothing to Compute the Probability of the Bigram (“machine learning”) Given a Small Corpus Where It Does Not Occur

✅ Problem:

You are given a small corpus where the bigram **("machine learning")** does **not occur**.

We need to apply **Laplace (Add-One) Smoothing** to estimate its probability.

🧰 Given Corpus (Example):

Assume this toy corpus with 6 sentences:

```
1. deep learning is powerful
2. machine intelligence is rising
3. learning algorithms improve systems
4. neural networks are deep
5. machine vision is improving
6. reinforcement learning is useful
```

📊 Step 1: Gather Counts

- Vocabulary (V) = number of unique words = **16**
- Count of “machine” = **2**
- Count of bigram “machine learning” = **0** (not present)

📌 Step 2: Laplace Smoothing Formula

$$P_{\text{Lap}}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + V}$$

Where:

- $C(w_{i-1}, w_i)$ = count of the bigram (“machine learning”)
- $C(w_{i-1})$ = count of the first word “machine”
- V = vocabulary size (number of unique words)

🧮 Step 3: Plug Values

$$P(\text{learning} \mid \text{machine}) = \frac{0+1}{2+16} = \frac{1}{18} \approx 0.0556$$

✅ This non-zero probability is thanks to **Laplace smoothing**, avoiding the zero-frequency problem.

🔍 Without Laplace:

$$P(\text{learning} \mid \text{machine}) = \frac{0}{2} = 0$$

Which would cause issues in downstream tasks like perplexity calculation, translation, etc.

🔹 Q13. Analyze the Limitations of Basic Smoothing Methods (e.g., Laplace, Add-k) in Handling Sparse Data

---

✅ What Are Basic Smoothing Techniques?

- **Laplace Smoothing (Add-One)**: Adds 1 to every count to avoid zero probability.
- **Add-k Smoothing**: A general form where a small constant $k \in (0, 1)$ is added instead of 1.

$$P_{\text{Add-k}}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i) + k}{C(w_{i-1}) + kV}$$

⚠️ Limitations of Laplace/Add-k:

1. Over-Penalizes Frequent Events

- Adds the same smoothing to all events.
- Frequent bigrams lose probability mass to rare or unseen bigrams.
- Makes the model **less sharp**, especially in large vocabularies.

2. Unrealistic Equal Treatment

- It assumes **all unseen bigrams are equally likely**, which is not true.
- For example, "banana airplane" and "machine learning" are treated equally if both are unseen.

3. Doesn't Use Contextual Diversity

- Doesn't take into account how diverse a word's usage is (which Kneser-Ney does via continuation probability).
- Can't distinguish between:
  - A rare word that occurs in many contexts (useful)
  - A rare word that occurs only once (less useful)

4. Poor Performance on Large Sparse Corpora

- In huge vocabularies, adding even 1 count becomes **mathematically imbalanced**.

- Probabilities for observed events get **unduly reduced**.

🧠 When to Use Laplace/Add-k?

✅ Useful in:

- Educational and demonstrative contexts
- Toy datasets or controlled environments
- Simple classifiers (e.g., Naïve Bayes)

❌ Not ideal for:

- Modern language modeling (e.g., GPT, BERT)
- Sparse real-world corpora

🧪 Alternative Techniques:

Use these instead for better performance:

- **Good-Turing Smoothing**: Redistributes probability mass based on frequency of frequencies.
- **Kneser-Ney Smoothing**: Considers how often a word appears in **different contexts** — state-of-the-art for n-gram models.
- **Backoff and Interpolation**: Combine higher-order and lower-order n-grams with learned weights.