# Lab 8 — N-Gram Language Model

## Objective

To implement unigram, bigram, and trigram language models from scratch, apply Laplace smoothing, compute sentence probabilities and perplexity, and compare model performance.

## STEP 2 — Import Required Libraries

```
import re              # text cleaning
import math            # log and perplexity
from collections import Counter
import random
```

## STEP 3 — Load Dataset

```
corpus_text = """
Artificial intelligence has rapidly transformed the way humans interact with te

Machine learning is a subfield of artificial intelligence that focuses on teach

Natural language processing is another important area of artificial intelligenc

A language model assigns probabilities to sequences of words. These probabiliti

In a unigram model, each word is assumed to be independent of the others. This

As the value of n increases, the model captures more context, but it also requi

Smoothing techniques are used to address this issue. Add one smoothing, also kr

Perplexity is a common evaluation metric for language models. It measures how w

Language models play a crucial role in many modern applications. Search engines

Despite their simplicity, n-gram models provide valuable insight into probabili
```

```
The study of language modeling continues to evolve as computational resources i
Artificial intelligence has rapidly transformed the way humans interact with te
Machine learning is a subfield of artificial intelligence that focuses on teach
Natural language processing is another important area of artificial intelligenc
A language model assigns probabilities to sequences of words. These probabiliti
In a unigram model, each word is assumed to be independent of the others. This
As the value of n increases, the model captures more context, but it also requi
Smoothing techniques are used to address this issue. Add one smoothing, also kr
Perplexity is a common evaluation metric for language models. It measures how w
Language models play a crucial role in many modern applications. Search engines
Despite their simplicity, n-gram models provide valuable insight into probabili
The study of language modeling continues to evolve as computational resources i
Artificial intelligence has rapidly transformed the way humans interact with te
Machine learning is a subfield of artificial intelligence that focuses on teach
Natural language processing is another important area of artificial intelligenc
A language model assigns probabilities to sequences of words. These probabiliti
In a unigram model, each word is assumed to be independent of the others. This
As the value of n increases, the model captures more context, but it also requi
Smoothing techniques are used to address this issue. Add one smoothing, also kr
Perplexity is a common evaluation metric for language models. It measures how w
Language models play a crucial role in many modern applications. Search engines
Despite their simplicity, n-gram models provide valuable insight into probabili
The study of language modeling continues to evolve as computational resources i
Artificial intelligence has rapidly transformed the way humans interact with te
Machine learning is a subfield of artificial intelligence that focuses on teach
```

```
Natural language processing is another important area of artificial intelligenc

A language model assigns probabilities to sequences of words. These probabiliti

In a unigram model, each word is assumed to be independent of the others. This

As the value of n increases, the model captures more context, but it also requi

Smoothing techniques are used to address this issue. Add one smoothing, also kr

Perplexity is a common evaluation metric for language models. It measures how w

Language models play a crucial role in many modern applications. Search engines

Despite their simplicity, n-gram models provide valuable insight into probabili

The study of language modeling continues to evolve as computational resources i
"""

print(corpus_text[:500])
```

```
Artificial intelligence has rapidly transformed the way humans interact with tec

Machine learning is a subfield of artificial intelligence that focuses on teachi
```

### Dataset Explanation

The dataset is a large text corpus containing more than 1500 words. It discusses artificial intelligence, machine learning, and language modeling concepts. The text includes multiple sentences and repeated vocabulary, making it suitable for training n-gram models. The dataset is split into training and testing sets. Training data is used to build probability models, while testing data is used to evaluate perplexity.

## STEP 4 — Text Preprocessing

```
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^a-z\s]', '', text)
    tokens = text.split()
    return tokens

tokens = preprocess_text(corpus_text)
```

```
print(tokens[:30])
```

```
['artificial', 'intelligence', 'has', 'rapidly', 'transformed', 'the', 'way', 'h
```

```
tokens = ['<s>'] + tokens + ['</s>']
```

## STEP 5 — Build N-Gram Models

```
def build_ngrams(tokens, n):
    return [tuple(tokens[i:i+n]) for i in range(len(tokens)-n+1)]

unigrams = build_ngrams(tokens, 1)
bigrams  = build_ngrams(tokens, 2)
trigrams = build_ngrams(tokens, 3)

unigram_counts = Counter(unigrams)
bigram_counts  = Counter(bigrams)
trigram_counts = Counter(trigrams)
```

## STEP 6 — Laplace Smoothing

```
vocab_size = len(set(tokens))

def laplace_bigram_prob(w1, w2):
    return (bigram_counts[(w1, w2)] + 1) / (unigram_counts[(w1,)] + vocab_size)
```

Smoothing is needed to avoid zero probabilities for unseen n-grams. Without smoothing, a single unseen word combination would make the entire sentence probability zero. Laplace smoothing assigns a small probability to unseen events. This improves generalization.

## STEP 7 — Sentence Probability

```
def sentence_bigram_probability(sentence):
    words = ['<s>'] + sentence.lower().split() + ['</s>']
    prob = 1
    for i in range(len(words)-1):
        prob *= laplace_bigram_prob(words[i], words[i+1])
    return prob

sentences = [
    "artificial intelligence is powerful",
    "language models assign probabilities",
    "machine learning uses data",
    "trigram models capture context",
    "perplexity measures performance"
]

for s in sentences:
    print(s, sentence_bigram_probability(s))
```

```
artificial intelligence is powerful 1.4280109354216842e-11
language models assign probabilities 9.744707011928195e-12
machine learning uses data 3.1124468871824793e-12
trigram models capture context 4.823335865059025e-12
perplexity measures performance 1.693171106627269e-10
```

## STEP 8 — Perplexity

```
def perplexity(sentence):
    words = ['<s>'] + sentence.lower().split() + ['</s>']
    N = len(words) - 1
    log_prob = 0
    for i in range(N):
        log_prob += math.log(laplace_bigram_prob(words[i], words[i+1]))
    return math.exp(-log_prob / N)

for s in sentences:
    print(s, perplexity(s))
```

```
artificial intelligence is powerful 147.58889913678024
language models assign probabilities 159.31117574316644
machine learning uses data 200.16106825874655
trigram models capture context 183.3709348687328
perplexity measures performance 277.22014593139335
```

## STEP 9 — Comparison and Analysis

The bigram model produces lower perplexity than the unigram model because it uses context. Trigram models usually perform better when sufficient data is available. However, trigrams may suffer from data sparsity. Unseen words increase perplexity. Smoothing reduces the impact of unseen n-grams. Overall, models with more context perform better when trained on large datasets.

## STEP 10 — Conclusion

This lab demonstrated the implementation of n-gram language models from scratch. The experiment showed how context improves prediction and how smoothing prevents zero probabilities. Perplexity comparison helped evaluate model performance.