ASSIGNMENT 6

Code:

import nltk

from nltk.util import ngrams

from nltk import word\_tokenize

from collections import defaultdict, Counter

import math

# Download the necessary NLTK data

nltk.download('punkt')

nltk.download('punkt\_tab') # Download the missing resource

class NGramModel:

    def \_\_init\_\_(self, n):

        self.n = n

        self.ngram\_counts = defaultdict(int)

        self.context\_counts = defaultdict(int)

        self.vocab = set()

    def train(self, text):

        tokens = ['<s>'] \* (self.n - 1) + word\_tokenize(text.lower()) + ['</s>']

        self.vocab.update(tokens)

        for ngram in ngrams(tokens, self.n):

            context = ngram[:-1]

            self.ngram\_counts[ngram] += 1

            self.context\_counts[context] += 1

    def calculate\_sentence\_probability(self, sentence):

        tokens = ['<s>'] \* (self.n - 1) + word\_tokenize(sentence.lower()) + ['</s>']

        prob = 0.0

        V = len(self.vocab)

        for ngram in ngrams(tokens, self.n):

            context = ngram[:-1]

            count\_ngram = self.ngram\_counts[ngram]

            count\_context = self.context\_counts[context]

            # Laplace smoothing

            smoothed\_prob = (count\_ngram + 1) / (count\_context + V)

            prob += math.log(smoothed\_prob)

        return math.exp(prob)

#bigram

text\_corpus = "The quick brown fox jumps over the lazy dog. The dog barked loudly."

model = NGramModel(n=2)

model.train(text\_corpus)

test\_sentence = "The dog barked"

probability = model.calculate\_sentence\_probability(test\_sentence)

print(f"Probability of '{test\_sentence}': {probability}")

Output:

Probability of 'The dog barked': 0.00017006802721088418