Demonstrate the N-gram modeling to analyze and establish the probability distribution across sentences and explore the utilization of unigrams, bigrams, and trigrams in diverse English sentences to illustrate the impact of varying n-gram orders on the calculated probabilities.

import nltk

import re

from nltk import word\_tokenize, ngrams

from collections import Counter

# Ensure the necessary NLTK data is downloaded

nltk.download('punkt')

# Sample text corpus

corpus = """The Arabian Knights.

These are the fairy tales of the east. The stories of the Arabian knights are translated in many languages"""

def tokenize(text):

text = re.sub(r'[^a-zA-Z\s]', '', text) # Remove non-alphabetic characters

words = text.lower().split() # Convert to lowercase and split by spaces

return words

# Tokenize the text (convert to lowercase for consistency)

tokens = tokenize(corpus)

def compute\_ngram\_probabilities(tokens, n):

"""

Computes n-gram probabilities using the conditional probability formula.

For n = 1 (unigrams):

P(word) = count(word) / total\_word\_count

For n > 1:

P(word\_n | word\_1, ..., word\_(n-1)) = count(word\_1, ..., word\_n) / count(word\_1, ..., word\_(n-1))

"""

# Generate the n-grams and count their frequencies

ngrams\_list = list(ngrams(tokens, n))

ngram\_counts = Counter(ngrams\_list)

if n == 1:

total\_count = sum(ngram\_counts.values())

probabilities = {ngram: count / total\_count for ngram, count in ngram\_counts.items()}

else:

# Count the (n-1)-gram prefixes

prefix\_counts = Counter(ngrams(tokens, n-1))

probabilities = {}

for ngram, count in ngram\_counts.items():

prefix = ngram[:-1] # All but the last word form the prefix

probabilities[ngram] = count / prefix\_counts[prefix]

return probabilities

unigram\_probs = compute\_ngram\_probabilities(tokens, 1)

bigram\_probs = compute\_ngram\_probabilities(tokens, 2)

trigram\_probs = compute\_ngram\_probabilities(tokens, 3)

# Display probabilities

print("Unigram Probabilities:")

for ngram, prob in unigram\_probs.items():

# Unigram is a single-element tuple

print(f"P({ngram[0]}) = {prob:.4f}")

print("\nBigram Probabilities:")

for ngram, prob in bigram\_probs.items():

# For bigrams, display as P(word2 | word1)

print(f"P({ngram[1]} | {ngram[0]}) = {prob:.4f}")

print("\nTrigram Probabilities:")

for ngram, prob in trigram\_probs.items():

# For trigrams, display as P(word3 | word1 word2)

print(f"P({ngram[2]} | {ngram[0]} {ngram[1]}) = {prob:.4f}")