Building an N-gram Model

Objective:

Develop a basic understanding of n-gram models and how to implement them using Python and the Natural Language Toolkit (NLTK).

Background:

An n-gram is a contiguous sequence of n items from a given sample of text or speech. They are widely used in Natural Language Processing (NLP) tasks, such as language modeling, text prediction, translation, and spell correction.

Dataset:

We will use the 'reuters' corpus from the NLTK library for this assignment. This corpus consists of 10,788 news documents totaling 1.3 million words. The documents have been classified into 90 topics, and thus the corpus is often used for experiments in text categorization.

Requirements (we will provide this already):

- Install the necessary libraries: Python, NLTK.
- Download the 'reuters' corpus from NLTK: nltk.download('reuters').

Tasks:

1. Load the 'reuters' corpus.
2. Preprocess the corpus: Tokenize the text into individual words, convert all words to lowercase, and remove punctuation and numbers.
3. Write a function that creates n-grams from the processed corpus (try for different n values, like 2, 3, 4).
4. Write a function that calculates and displays the frequency of each n-gram in the corpus.=
5. Implement a simple predictive text model: Given an input sequence of words, predict the next most likely word based on the highest n-gram frequency.

Hints:

- Use the ngrams function from NLTK to generate n-grams: ngrams(processed_corpus, n).

- Use the FreqDist function from NLTK to calculate the frequency of each n-gram: FreqDist(ngrams).