Experiment 3: NGram Model

N-Gram Models are statistical language models that capture the likelihood of a sequence of words in a text. The 'N' in N-Gram represents the number of words in each sequence. Common N-Grams include unigrams (1-word sequences), bigrams (2-word sequences), and trigrams (3-word sequences). These models are widely used in various NLP tasks such as language modeling, text analysis, and information retrieval.

N-Gram Probability Estimation:

To build an N-Gram Model, we estimate the probabilities of word sequences occurring in a given corpus. For example, in a bigram model, the probability of a word depends on the preceding word. These probabilities can be estimated using maximum likelihood estimation (MLE) or smoothing techniques like Laplace smoothing. These estimated probabilities provide insights into the likelihood of observing specific word sequences in a given language or context.

N-Gram Models find applications in:

1. Language Modeling: Predicting the probability of a word given its context, which is essential in speech recognition and natural language understanding.
2. Text Analysis: Identifying patterns and relationships in text data, facilitating tasks like sentiment analysis and information extraction.
3. Information Retrieval: Improving search engine algorithms by considering the likelihood of word sequences in documents and queries.
4. Speech Processing: Modeling phonemes and phonetic sequences for speech recognition and synthesis.
5. Spelling and Grammar Correction: Identifying errors in text by comparing observed N-Grams to expected N-Gram probabilities.
6. Keyword Prediction: Predicting the next word in a user's search query or text message, aiding in autocomplete and predictive text features.

Understanding N-Gram Models is fundamental for numerous NLP applications, providing valuable insights into the structure and patterns of natural language data.

Example:

Consider a practical scenario where we apply a bigram model to a corpus of medical records for information extraction rather than text generation. Suppose we want to identify medical conditions mentioned in patient records. The bigram model can help us achieve this by analyzing word sequences.

In the patient record corpus, we find the following bigram probabilities:

1) P(diagnosed | Patient) = 0.85

2) P(with | diagnosed) = 0.70

3) P(diabetes | with) = 0.60

4) P(hypertension | with) = 0.40

Now, if we encounter the phrase "Patient diagnosed with," we can use the bigram model to predict the medical condition mentioned next. In this case, "diabetes" has the highest probability, so we can extract the information that the patient was diagnosed with diabetes.

Algorithm:

1. Import Libraries:

    • Import the required libraries, including NLTK and its components for tokenization, n-gram generation, and part-of-speech tagging.

2. Download NLTK Data:

    • Ensure that the necessary NLTK data is downloaded using nltk.download('punkt') and nltk.download('averaged_perceptron_tagger') if not already downloaded.

3. Tokenization and Basic Text Preprocessing:

    • Tokenize the input text into words using word_tokenize.

    • Convert words to lowercase and remove punctuation.

    • Remove stopwords using NLTK's stopwords list.

4. Generate N-grams:

    • Create a function generate_ngrams to generate n-grams of different sizes (unigrams, bigrams, trigrams, etc.).

    • Iterate over various n-gram sizes (e.g., 1 to 4).

    • Generate n-grams from the preprocessed text and print the n-grams for each size.

5. Part-of-Speech Tagging:

    • Tokenize the text into sentences and words.

    • Create a function get_word_pos to obtain the part-of-speech tag for a word.

- Create a function classify_word to classify a word as a noun, pronoun, adjective, or other based on its part-of-speech tag.
- Create a table to store word classifications.

6. Word Classification:
   - Process each word in the tokenized text.
   - For each word, determine its part-of-speech tag and classify it.
   - Store the word and its classification in the word table.
7. Print Results:
   - Print a table showing the word classifications, including the word and its classification (e.g., Noun, Pronoun, Adjective, or Other).