Practical No:3

Title:
Demonstrate a bigram / trigram language model. Generate regular expressions for a given text.

Aim:

1. To demonstrate how to build bigram and trigram language models.
2. To generate a regular expression for a given text.

Pre-requisites:

1. Basic understanding of NLP concepts, especially **n-gram models**.
2. Familiarity with probability theory and how n-gram models are used in predicting sequences.
3. Knowledge of regular expressions and their role in pattern matching.
4. Familiarity with Python and NLP libraries such as `nltk`.

Theory:

Part 1: N-gram Language Models (Bigram and Trigram)

An **N-gram model** is a type of probabilistic language model used to predict the next word in a sequence, given the previous N-1 words. It assumes that the probability of a word depends only on the previous words (up to N-1).

- **Unigram Model:** The probability of each word depends only on itself.
- **Bigram Model (2-gram):** The probability of a word depends on the previous word.

        - Example: In the sentence, “I love coding,” a bigram model breaks it into pairs: `(I, love)`, `(love, coding)`.
- **Trigram Model (3-gram):** The probability of a word depends on the previous two words.
    - Example: “I love coding” is broken into trigrams: `(I, love, coding)`.

Part 2: Regular Expressions

A **regular expression (regex)** is a sequence of characters that define a search pattern. Regular expressions are used for string matching, text search, and text manipulation.

Common regex patterns include:

- **Character Classes:** `[a-zA-Z]` matches any letter.
- **Quantifiers:** `*` (0 or more), `+` (1 or more), `?` (0 or 1).
- **Anchors:** `^` (start of string), `$` (end of string).

Steps for Bigram/Trigram Model:

1. **Text Preprocessing:**
    - Convert the text to lowercase.
    - Tokenize the text into words.
    - Remove stop words or punctuation (optional).
2. **Build the Bigram/Trigram Model:**
    - Use a bigram or trigram model to calculate word pair or triplet frequencies.
    - Calculate conditional probabilities of words appearing after one or two words.
3. **Generate Sentence:**

   - Starting with a word, use the bigram or trigram model to predict the next word based on probability.

Steps for Regular Expression Generation:

1. **Input a Text:**
   - Extract key patterns, such as email addresses, phone numbers, or specific word sequences.
2. **Create Regex Pattern:**
   - Write regular expressions that match specific patterns in the text.

Conclusion:

Bigram and trigram models are important for capturing the local dependencies between words in text, making them useful for tasks like sentence generation or next-word prediction. Regular expressions are powerful tools for extracting patterns such as email addresses or phone numbers from text, making them invaluable in text processing tasks. Together, these techniques are foundational in many NLP applications.