Experiment: 7

AIM: Write a Program on Tri-grams.

OBJECTIVES: To understand and implement **tri-grams**, which are pairs of consecutive words in a given text.

REQUIREMENTS: Python (version 3.x or above), NLTK library.

THEORY:

In Language modeling is the way of determining the probability of any sequence of words. Language modeling is used in various applications such as Speech Recognition, Spam filtering, etc. Language modeling is the key aim behind implementing many state-of-the-art Natural Language Processing models.

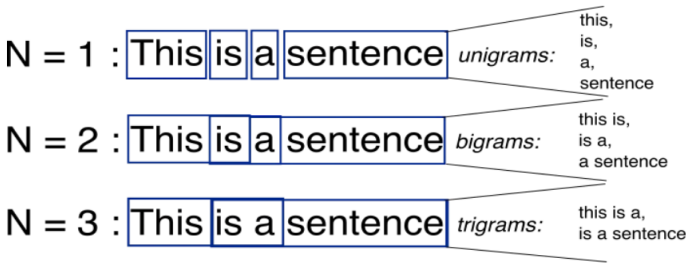
Tri-gram is a sequence of three consecutive words from a text. Trigrams help in understanding the context and word relationships better than unigrams and bigrams.

Example:

Sentence: "Machine learning is awesome"

Tri-grams:

- ("Machine", "learning", "is")
- ("learning", "is", "awesome")



Implementing Tri-grams using NLTK library:

Step 1: Importing necessary libraries like nltk and using ngrams to perform Tri gram.

```
import nltk
from nltk.util import ngrams
from nltk.tokenize import word_tokenize
# Download required tokenizer models
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\tdhan\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

Step 2: Taking Input text and dividing them in form of tokens.

```
# Sample input text
text = "Natural Language Processing with Python is powerful and \
interesting to train Various Machine learning and Deep learning models."
# Tokenize the text
tokens = word_tokenize(text)
print("Word Tokens:", tokens)
```

```
Word Tokens: ['Natural', 'Language', 'Processing', 'with', 'Python', 'is', 'powerful', 'and', 'interesting',
'Various', 'Machine', 'learning', 'and', 'Deep', 'learning', 'models', '.']
```

Step 3: Generating Tri-grams using the above text data and set the value of n = 3.

```
# Generate trigrams
tri_gram_result = list(ngrams(tokens,3))
# Display the trigrams
print("Tri-grams:")
for triplet in tri_gram_result:
    print(triplet)
```

```
Tri-grams:
('Natural', 'Language', 'Processing')
('Language', 'Processing', 'with')
('Processing', 'with', 'Python')
('with', 'Python', 'is')
('Python', 'is', 'powerful')
('is', 'powerful', 'and')
('powerful', 'and', 'interesting')
('and', 'interesting', 'to')
('interesting', 'to', 'train')
('to', 'train', 'Various')
('train', 'Various', 'Machine')
('Various', 'Machine', 'learning')
('Machine', 'learning', 'and')
('learning', 'and', 'Deep')
('and', 'Deep', 'learning')
('Deep', 'learning', 'models')
('learning', 'models', '.')
```

Conclusion

In this experiment, we successfully implemented tri-gram generation using the NLTK library in Python. Tri-grams provide deeper contextual understanding compared to unigrams and bigrams and are widely used in various NLP applications like language modeling, chatbots, and machine translation.