Ex.no:9 **TAGGING WORDS N-GRAMS**
DATE:24/01/2024

AIM:
To extract and identify meaningful bigram (two-word) and trigram (three-word) collocations from a given corpus of text using Python and the Natural Language Toolkit (nltk)

PROCEDURE:
Install and Import Necessary Libraries:

- Ensure nltk is installed in your Python environment.
- Import modules for tokenization and collocation finding from nltk.

Load Corpus Data:

- Use a prebuilt corpus from nltk (e.g., Jane Austen's "Emma") or load your custom text data.

Tokenize the Text:

- Convert the corpus into individual words using a tokenizer.
- Clean the tokens by removing punctuation and converting words to lowercase.

Find Bigrams:

- Use BigramCollocationFinder from nltk to identify pairs of words (bigrams).
- Apply frequency filters to remove less frequent bigrams.
- Use statistical measures like the likelihood ratio to rank and extract meaningful bigrams.

Find Trigrams:

- Use TrigramCollocationFinder from nltk to identify triplets of words (trigrams).
- Apply frequency filters to remove less frequent trigrams.
- Use statistical measures like the likelihood ratio to rank and extract meaningful trigrams.

CODE:

```
import nltk

from nltk.corpus import gutenberg

from nltk.collocations import BigramCollocationFinder, TrigramCollocationFinder

from nltk.metrics import BigramAssocMeasures, TrigramAssocMeasures


# Download necessary nltk data

nltk.download('gutenberg')

nltk.download('punkt')


# Load sample corpus data
```

```
corpus = gutenberg.raw('austen-emma.txt')  # Jane Austen's "Emma"

# Tokenize the corpus into words
tokens = nltk.word_tokenize(corpus)

# Filter tokens to remove punctuation and lowercase
filtered_tokens = [word.lower() for word in tokens if word.isalpha()]

# Create a BigramCollocationFinder
bigram_finder = BigramCollocationFinder.from_words(filtered_tokens)
bigram_finder.apply_freq_filter(5)  # Filter out bigrams that occur less than 5 times

# Extract top 10 bigrams based on their likelihood ratio
top_bigrams = bigram_finder.nbest(BigramAssocMeasures.likelihood_ratio, 10)

# Create a TrigramCollocationFinder
trigram_finder = TrigramCollocationFinder.from_words(filtered_tokens)
trigram_finder.apply_freq_filter(3)  # Filter out trigrams that occur less than 3 times

# Extract top 10 trigrams based on their likelihood ratio
top_trigrams = trigram_finder.nbest(TrigramAssocMeasures.likelihood_ratio, 10)

# Print results
print("Top 10 Bigrams:")
for bigram in top_bigrams:
    print(bigram)

print("\nTop 10 Trigrams:")
for trigram in top_trigrams:
    print(trigram)
```

OUTPUT:

```
[nltk_data] Downloading package gutenberg to
[nltk_data]     C:\Users\Praveena\AppData\Roaming\nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Praveena\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Top 10 Bigrams:
('i', 'am')
('had', 'been')
('to', 'be')
('frank', 'churchill')
('it', 'was')
('miss', 'woodhouse')
('have', 'been')
('could', 'not')
('any', 'thing')
('my', 'dear')

Top 10 Trigrams:
('i', 'am', 'not')
('i', 'am', 'sure')
('the', 'sort', 'of')
('the', 'whole', 'of')
('the', 'subject', 'of')
('the', 'rest', 'of')
('the', 'idea', 'of')
('the', 'part', 'of')
('the', 'evening', 'of')
('the', 'degree', 'of')
```