In [5]:
```
import nltk
```

In [7]:
```
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```

Out[7]:
```
True
```

In [14]:
```
import nltk
nltk.data.path.append("/path/to/nltk_data")
nltk.download("punkt")
```

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

Out[14]:
```
True
```

In [15]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk import pos_tag

nltk.download('punkt')

def word_analysis(text):
    # Tokenization
    tokens = word_tokenize(text)
    print("Tokens:", tokens)

    # Stemming
    stemmer = PorterStemmer()
    stemmed_words = [stemmer.stem(token) for token in tokens]
    print("Stemmed Words:", stemmed_words)

    # Part-of-Speech Tagging
    pos_tags = pos_tag(tokens)
    print("POS Tags:", pos_tags)

# Example text
text_to_analyze = "Natural Language Processing is fascinating!"

# Perform word analysis
word_analysis(text_to_analyze)
```

```
[nltk_data] Downloading package punkt to C:\Users\maddela
[nltk_data]     saikiran\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Tokens: ['Natural', 'Language', 'Processing', 'is', 'fascinating', '!']
Stemmed Words: ['natur', 'languag', 'process', 'is', 'fascin', '!']
POS Tags: [('Natural', 'JJ'), ('Language', 'NNP'), ('Processing', 'NNP'), ('is',
'VBZ'), ('fascinating', 'VBG'), ('!', '.')]
```

In [16]:
```
import random
from collections import defaultdict

class WordGenerator:
    def __init__(self, n=2):
        self.n = n  # Order of the Markov chain
        self.model = defaultdict(list)

    def train(self, corpus):
        for word in corpus:
            word = word.lower()
            for i in range(len(word) - self.n):
                prefix = word[i:i + self.n]
                next_char = word[i + self.n]
                self.model[prefix].append(next_char)

    def generate_word(self, length=5, seed=None):
        if seed is None:
            seed = random.choice(list(self.model.keys()))
        else:
            seed = seed.lower()

        current_prefix = seed[-self.n:]

        generated_word = seed

        for _ in range(length - self.n):
            next_char = random.choice(self.model.get(current_prefix, ['']))
            generated_word += next_char
            current_prefix = generated_word[-self.n:]

        return generated_word

# Example usage
corpus = ["apple", "banana", "cherry", "grape", "orange", "pear"]

word_generator = WordGenerator(n=2)
word_generator.train(corpus)

generated_word = word_generator.generate_word(length=8)
print("Generated Word:", generated_word)
```

```
Generated Word: ear
```

In [17]:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer

nltk.download('punkt')
nltk.download('wordnet')

def morphology_analysis(text):
    # Tokenization
    tokens = word_tokenize(text)
    print("Tokens:", tokens)

    # Stemming
    stemmer = PorterStemmer()
    stemmed_words = [stemmer.stem(token) for token in tokens]
    print("Stemmed Words:", stemmed_words)

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    lemmatized_words = [lemmatizer.lemmatize(token) for token in tokens]
    print("Lemmatized Words:", lemmatized_words)

# Example text
text_to_analyze = "Natural Language Processing involves the study of linguistic str

# Perform morphology analysis
morphology_analysis(text_to_analyze)
```

```
[nltk_data] Downloading package punkt to C:\Users\maddela
[nltk_data]     saikiran\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to C:\Users\maddela
[nltk_data]     saikiran\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Tokens: ['Natural', 'Language', 'Processing', 'involves', 'the', 'study', 'of', 'l
inguistic', 'structures', '.']
Stemmed Words: ['natur', 'languag', 'process', 'involv', 'the', 'studi', 'of', 'li
nguist', 'structur', '.']
Lemmatized Words: ['Natural', 'Language', 'Processing', 'involves', 'the', 'stud
y', 'of', 'linguistic', 'structure', '.']
```

In [18]:
```python
import nltk
from nltk import ngrams
from nltk.tokenize import word_tokenize

nltk.download('punkt')

def generate_ngrams(text, n):
    # Tokenize the text into words
    words = word_tokenize(text)

    # Generate n-grams
    n_grams = list(ngrams(words, n))

    return n_grams

# Example text
text_to_analyze = "Natural Language Processing is fascinating and powerful."

# Specify the value of 'n' for n-grams
n_value = 3

# Generate n-grams
result_ngrams = generate_ngrams(text_to_analyze, n_value)

# Print the result
print(f"{n_value}-grams:", result_ngrams)
```

```
3-grams: [('Natural', 'Language', 'Processing'), ('Language', 'Processing', 'is'),
('Processing', 'is', 'fascinating'), ('is', 'fascinating', 'and'), ('fascinating',
'and', 'powerful'), ('and', 'powerful', '.')]
[nltk_data] Downloading package punkt to C:\Users\maddela
[nltk_data]     saikiran\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [19]:
```python
import nltk
from nltk import ngrams
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist

nltk.download('punkt')

def add_one_smoothing_ngrams(text, n):
    # Tokenize the text into words
    words = word_tokenize(text)

    # Generate n-grams
    n_grams = list(ngrams(words, n))

    # Calculate frequencies of n-grams
    freq_dist = FreqDist(n_grams)
    # Vocabulary size (unique n-grams)
    vocab_size = len(set(n_grams))

    # Add-one smoothing
    smoothed_n_grams = [(gram, (freq_dist[gram] + 1) / (len(n_grams) + vocab_size))

    return smoothed_n_grams

# Example text
text_to_analyze = "Natural Language Processing is fascinating and powerful."

# Specify the value of 'n' for n-grams
n_value = 2

# Generate n-grams with add-one smoothing
result_ngrams = add_one_smoothing_ngrams(text_to_analyze, n_value)

# Print the result
print(f"{n_value}-grams with Add-One Smoothing:", result_ngrams)
```

```
2-grams with Add-One Smoothing: [(('Natural', 'Language'), 0.14285714285714285),
(('Language', 'Processing'), 0.14285714285714285), (('Processing', 'is'), 0.142857
14285714285), (('is', 'fascinating'), 0.14285714285714285), (('fascinating', 'an
d'), 0.14285714285714285), (('and', 'powerful'), 0.14285714285714285), (('powerfu
l', '.'), 0.14285714285714285)]
```

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

In [20]:
```
import nltk
from nltk.tokenize import word_tokenize

nltk.download('punkt')

def word_tokenizer(text):
    # Tokenize the text into words
    tokens = word_tokenize(text)

    return tokens

# Example text
text_to_tokenize = "Word tokenization is an essential step in natural language proc

# Tokenize the text
tokenized_words = word_tokenizer(text_to_tokenize)

# Print the result
print("Tokenized Words:", tokenized_words)
```

```
Tokenized Words: ['Word', 'tokenization', 'is', 'an', 'essential', 'step', 'in',
'natural', 'language', 'processing', '.']
```

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

In [21]:
```
import nltk
from nltk.tokenize import sent_tokenize

nltk.download('punkt')

def sentence_tokenizer(text):
    # Tokenize the text into sentences
    sentences = sent_tokenize(text)
    return sentences

# Example text
text_to_tokenize = "Sentence tokenization is important. It helps break text into me

# Tokenize the text into sentences
tokenized_sentences = sentence_tokenizer(text_to_tokenize)

# Print the result
print("Tokenized Sentences:")
for i, sentence in enumerate(tokenized_sentences, start=1):
    print(f"{i}. {sentence}")
```

```
Tokenized Sentences:
1. Sentence tokenization is important.
2. It helps break text into meaningful sentences.
3. NLTK provides tools for this task.
```

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

In [22]:

```
def paragraph_tokenizer(text):
    # Split the text into paragraphs based on newline characters
    paragraphs = text.split('\n\n')  # Adjust as needed for your specific text form

    return paragraphs

# Example text
text_to_tokenize = """
Paragraph 1: NLTK is a powerful library for natural language processing.
It provides tools for tasks such as tokenization, stemming, and part-of-speech tagg

Paragraph 2: The library is widely used in the field of artificial intelligence.
It is a valuable resource for researchers and developers working on NLP projects.
"""

# Tokenize the text into paragraphs
tokenized_paragraphs = paragraph_tokenizer(text_to_tokenize)

# Print the result
print("Tokenized Paragraphs:")
for i, paragraph in enumerate(tokenized_paragraphs, start=1):
    print(f"{i}. {paragraph.strip()}")
```

```
Tokenized Paragraphs:
1. Paragraph 1: NLTK is a powerful library for natural language processing.
It provides tools for tasks such as tokenization, stemming, and part-of-speech tag
ging.
2. Paragraph 2: The library is widely used in the field of artificial intelligenc
e.
It is a valuable resource for researchers and developers working on NLP projects.
```

In [23]:

```
import nltk
from nltk.corpus import reuters

nltk.download('reuters')

def load_reuters_corpus():
    # Load the Reuters corpus from NLTK
    corpus = reuters

    # Display some basic information about the corpus
    print("Number of Categories:", len(corpus.categories()))
    print("Categories:", corpus.categories()[:10])
    print("File IDs in 'crude' category:", corpus.fileids('crude')[:5])

    # Access and print the text of a specific document
    document_id = 'test/14826'
    document_text = corpus.raw(document_id)
    print("\nText of Document (ID:", document_id, "):\n", document_text[:500])

# Load and explore the Reuters corpus
load_reuters_corpus()
```

```
[nltk_data] Downloading package reuters to C:\Users\maddela
[nltk_data]     saikiran\AppData\Roaming\nltk_data...
[nltk_data]   Package reuters is already up-to-date!
Number of Categories: 90
Categories: ['acq', 'alum', 'barley', 'bop', 'carcass', 'castor-oil', 'cocoa', 'co
conut', 'coconut-oil', 'coffee']
File IDs in 'crude' category: ['test/14829', 'test/15063', 'test/15200', 'test/152
30', 'test/15238']

Text of Document (ID: test/14826 ):
 ASIAN EXPORTERS FEAR DAMAGE FROM U.S.-JAPAN RIFT
  Mounting trade friction between the
  U.S. And Japan has raised fears among many of Asia's exporting
  nations that the row could inflict far-reaching economic
  damage, businessmen and officials said.
      They told Reuter correspondents in Asian capitals a U.S.
  Move against Japan might boost protectionist sentiment in the
  U.S. And lead to curbs on American imports of their products.
      But some exporters said that while the conflict wo
```

```
In [ ]:
```