1.Unigrams

In [8]:
```
from collections import Counter

def get_unigrams(text):
    # Tokenize the text into words
    words = text.split()
    # Count the frequency of each word
    unigrams = Counter(words)
    return unigrams

# Example usage
text = "this is a sample text with sample words"
unigrams = get_unigrams(text)
print(unigrams)
```

```
Counter({'sample': 2, 'this': 1, 'is': 1, 'a': 1, 'text': 1, 'with': 1, 'words': 1})
```

2. Bigrams

In [9]:
```
from collections import Counter
from nltk import bigrams

def get_bigrams(text):
    # Tokenize the text into words
    words = text.split()
    # Create bigrams from the words
    bigrams_list = list(bigrams(words))
    # Count the frequency of each bigram
    bigrams_count = Counter(bigrams_list)
    return bigrams_count

# Example usage
text = "this is a sample text with sample words"
bigrams = get_bigrams(text)
print(bigrams)
```

```
Counter({('this', 'is'): 1, ('is', 'a'): 1, ('a', 'sample'): 1, ('sample', 'text'): 1, ('text', 'with'): 1, ('with', 'sample'): 1, ('sample', 'words')
```

3. Trigrams

In [10]:
```
from collections import Counter
from nltk import trigrams

def get_trigrams(text):
    # Tokenize the text into words
    words = text.split()
    # Create trigrams from the words
    trigrams_list = list(trigrams(words))
    # Count the frequency of each trigram
    trigrams_count = Counter(trigrams_list)
    return trigrams_count

# Example usage
text = "this is a sample text with sample words"
trigrams = get_trigrams(text)
print(trigrams)
```

```
Counter({('this', 'is', 'a'): 1, ('is', 'a', 'sample'): 1, ('a', 'sample', 'text'): 1, ('sample', 'text', 'with'): 1, ('text', 'with', 'sample'): 1, (
```

4. Bigram Probabilities

In [13]:
```
from collections import Counter, defaultdict
from nltk import bigrams
```

```
def get_bigram_probabilities(text):
    # Tokenize the text into words
    words = text.split()
    # Create bigrams from the words
    bigrams_list = list(bigrams(words))
    # Count the frequency of each bigram
    bigram_freq = Counter(bigrams_list)
    # Count the frequency of each word (unigram)
    unigram_freq = Counter(words)

    # Calculate bigram probabilities
    bigram_probabilities = defaultdict(float)
    for bigram in bigram_freq:
        bigram_probabilities[bigram] = bigram_freq[bigram] / unigram_freq[bigram[0]]

    return bigram_probabilities

# Example usage
text = "this is a sample text with sample words"
bigram_probabilities = get_bigram_probabilities(text)
print(bigram_probabilities)
```

```
defaultdict(<class 'float'>, {('this', 'is'): 1.0, ('is', 'a'): 1.0, ('a', 'sample'): 1.0, ('sample', 'text'): 0.5, ('text', 'with'): 1.0, ('with', 's
```

5. Next Word Prediction

In [14]:

```
def predict_next_word(text, word):
    # Get the bigram probabilities
    bigram_probabilities = get_bigram_probabilities(text)

    # Filter bigrams that start with the given word
    candidates = {bigram[1]: prob for bigram, prob in bigram_probabilities.items() if bigram[0] == word}

    if not candidates:
        return None

    # Predict the next word as the one with the highest probability
    next_word = max(candidates, key=candidates.get)
    return next_word

# Example usage
text = "this is a sample text with sample words"
word = "sample"
next_word = predict_next_word(text, word)
print(next_word)
```

```
text
```