```
import nltk
from nltk import ngrams
from collections import Counter
nltk.download('punkt')
text = """I like to learn natural language processing and also to learn machine learning."
tokens = nltk.word_tokenize(text)
trigrams = list(ngrams(tokens, 3))
trigram_freq = Counter(trigrams)
for trigram, freq in trigram_freq.items():
  print(f"{trigram}: {freq}")
```

```
('I', 'like', 'to'): 1
('like', 'to', 'learn'): 1
('to', 'learn', 'natural'): 1
('learn', 'natural', 'language'): 1
('natural', 'language', 'processing'): 1
('language', 'processing', 'and'): 1
('processing', 'and', 'also'): 1
('and', 'also', 'to'): 1
('also', 'to', 'learn'): 1
('to', 'learn', 'machine'): 1
('learn', 'machine', 'learning'): 1
('machine', 'learning', '.'): 1
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
import nltk
from nltk.util import ngrams
from collections import defaultdict
import random
nltk.download('punkt')
corpus = [
"I love to learn natural language processing",
"and also to learn machine learning"
]

tokenized_corpus = [nltk.word_tokenize(sentence.lower()) for sentence in corpus]
n = 2
n_grams = [ngrams(sentence, n) for sentence in tokenized_corpus]


next_word_dict = defaultdict(list)
for sentence in n_grams:
   for n_gram in sentence:
       prefix, next_word = n_gram[0], n_gram[1]
       next_word_dict[prefix].append(next_word)
def predict_next_word(prefix):
       prefix = prefix.lower()
       if prefix in next_word_dict:
             return random.choice(next_word_dict[prefix])
       else:
            return "No prediction available."
input_prefix = "I love"
```

```
predicted_word = predict_next_word(input_prefix)
print(f"Next word prediction for '{input_prefix}': {predicted_word}")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
Next word prediction for 'I love': No prediction available.
```

```
import nltk
from nltk import ngrams
from collections import Counter
nltk.download('punkt')
text="""I like to learn natural language processing and also to learn machine learning. ""
tokens = nltk.word_tokenize(text)
bigrams = list(ngrams(tokens, 2))
bigram_freq = Counter(bigrams)
word_freq = Counter(tokens)
bigram_probabilities = {}
for (w1, w2), count in bigram_freq.items():
   prob = count / word_freq[w1]
   bigram_probabilities[(w1, w2)] = prob
for bigram, prob in bigram_probabilities.items():
     print(f"P({bigram[1]} | {bigram[0]}) = {prob:.4f}")
```

```
P(like | I) = 1.0000
P(to | like) = 1.0000
P(learn | to) = 1.0000
P(natural | learn) = 0.5000
P(language | natural) = 1.0000
P(processing | language) = 1.0000
P(and | processing) = 1.0000
P(also | and) = 1.0000
P(to | also) = 1.0000
P(machine | learn) = 0.5000
P(learning | machine) = 1.0000
P(. | learning) = 1.0000
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
import nltk
from collections import Counter
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.util import bigrams
nltk.download('punkt')
nltk.download('stopwords')
text_corpus = """Natural language processing (NLP) is a subfield of␣
linguistics, computer science, and artificial intelligence"""
tokens = word_tokenize(text_corpus)
tokens = [token.lower() for token in tokens]
stop_words = set(stopwords.words('english'))
tokens = [token for token in tokens if token.isalpha() and token not in
stop_words]
bigrams_list = list(bigrams(tokens))
bigram_counts = Counter(bigrams_list)
print(bigram_counts)
```

```
Counter({('natural', 'language'): 1, ('language', 'processing'): 1, ('processin
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
import nltk
from nltk.tokenize import word_tokenize
from collections import Counter

# Ensure the required nltk data is downloaded
nltk.download('punkt')

# Sample text corpus
text_corpus = """I like to learn natural language processing and also to learn machine lea
"""

# Tokenize the text to get unigrams
tokens = word_tokenize(text_corpus)

# Count the frequency of each unigram
unigram_counts = Counter(tokens)

# Print the unigrams and their counts
for unigram, count in unigram_counts.items():
    print(f"{unigram}: {count}")
```

```
I: 1
like: 1
to: 2
learn: 2
natural: 1
language: 1
processing: 1
and: 1
also: 1
machine: 1
learning: 1
.: 1
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```