```
import nltk
from nltk import bigrams, trigrams, ngrams, FreqDist
from nltk.probability import ConditionalFreqDist
from nltk.util import ngrams
from nltk.corpus import stopwords
import pandas as pd
```

```
nltk.download('punkt')
nltk.download('stopwords')
```

```
[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!
True
```

with nltk library

```
# Sample corpus text
corpus_text = "The cat Tom chased the mouse Jerry, and the mouse Jerry ran away from the cat Tom."

# Get NLTK stop words
stop_words = set(stopwords.words("english"))

# Tokenize the text into words
words = nltk.word_tokenize(corpus_text)

# Remove stop words
words = [word for word in words if word.lower() not in stop_words]

# Create unigrams, bigrams, and trigrams
unigrams = list(ngrams(words, 1))
bigrams = list(ngrams(words, 2))
trigrams = list(ngrams(words, 3))

# Display the lists of unigrams, bigrams, and trigrams
print("Unigrams:", unigrams)
print("Bigrams:", bigrams)
print("Trigrams:", trigrams)
```

```
Unigrams: [('cat',), ('Tom',), ('chased',), ('mouse',), ('Jerry',), (',',), ('mouse',), ('Jerry',), ('ran',), ('away',), ('cat',), ('Tom',), (
Bigrams: [('cat', 'Tom'), ('Tom', 'chased'), ('chased', 'mouse'), ('mouse', 'Jerry'), ('Jerry', ','), (',', 'mouse'), ('mouse', 'Jerry'), ('Je
Trigrams: [('cat', 'Tom', 'chased'), ('Tom', 'chased', 'mouse'), ('chased', 'mouse', 'Jerry'), ('mouse', 'Jerry', ','), ('Jerry', ',', 'mouse'
```

```
# Calculate bigram frequencies and probabilities
bigram_freq = FreqDist(bigrams)
total_bigrams = len(bigrams)

# Create a conditional frequency distribution for bigrams
cfd = ConditionalFreqDist(bigrams)

# Display bigram counts and probabilities
print("\nBigram Counts:")
for bigram, freq in bigram_freq.items():
    print(f"{bigram}: {freq}")

print("\nBigram Probabilities:")
for bigram, freq in bigram_freq.items():
    probability = freq / total_bigrams
    print(f"{bigram}: {probability:.4f}")
```

```
Bigram Counts:
('cat', 'Tom'): 2
('Tom', 'chased'): 1
('chased', 'mouse'): 1
('mouse', 'Jerry'): 2
('Jerry', ','): 1
(',', 'mouse'): 1
('Jerry', 'ran'): 1
('ran', 'away'): 1
('away', 'cat'): 1
('Tom', '.'): 1

Bigram Probabilities:
('cat', 'Tom'): 0.1667
('Tom', 'chased'): 0.0833
('chased', 'mouse'): 0.0833
('mouse', 'Jerry'): 0.1667
('Jerry', ','): 0.0833
(',', 'mouse'): 0.0833
('Jerry', 'ran'): 0.0833
('ran', 'away'): 0.0833
('away', 'cat'): 0.0833
('Tom', '.'): 0.0833
```

```
df = pd.DataFrame(bigram_freq.items(), columns=["bigram", "count"])
df
```

	bigram	count
0	(cat, Tom)	2
1	(Tom, chased)	1
2	(chased, mouse)	1
3	(mouse, Jerry)	2
4	(Jerry, ,)	1
5	(,, mouse)	1
6	(Jerry, ran)	1
7	(ran, away)	1
8	(away, cat)	1
9	(Tom, .)	1

```
bigram_probabilities = [ [ bigram, freq / total_bigrams ] for bigram, freq in bigram_freq.items() ]
df = pd.DataFrame( bigram_probabilities, columns=["bigram", "count"])
df
```

	bigram	count
0	(cat, Tom)	0.166667
1	(Tom, chased)	0.083333
2	(chased, mouse)	0.083333
3	(mouse, Jerry)	0.166667
4	(Jerry, ,)	0.083333
5	(,, mouse)	0.083333
6	(Jerry, ran)	0.083333
7	(ran, away)	0.083333
8	(away, cat)	0.083333
9	(Tom, .)	0.083333

without nltk library

```
def generate_ngrams(text, n):
    # Get NLTK stop words
    stop_words = set(stopwords.words("english"))

    # Tokenize the text into words
    words = text.split()

    # Remove stop words
    words = [word for word in words if word.lower() not in stop_words]

    # Create n-grams
    ngrams_list = []
    for i in range(len(words) - n + 1):
        ngram = tuple(words[i:i + n])
        ngrams_list.append(ngram)

    return ngrams_list
```

```
# Example usage:
text = corpus_text
unigrams = generate_ngrams(text, 1)
bigrams = generate_ngrams(text, 2)
trigrams = generate_ngrams(text, 3)

print("Unigrams:", unigrams)
print("Bigrams:", bigrams)
print("Trigrams:", trigrams)
```

```
('chased',), ('mouse',), ('Jerry,',), ('mouse',), ('Jerry',), ('ran',), ('away',), ('cat',), ('Tom.',)]
', 'chased'), ('chased', 'mouse'), ('mouse', 'Jerry,'), ('Jerry,', 'mouse'), ('mouse', 'Jerry'), ('Jerry', 'ran'), ('ran', 'away'), ('away', 'c
ed'), ('Tom', 'chased', 'mouse'), ('chased', 'mouse', 'Jerry,'), ('mouse', 'Jerry,', 'mouse'), ('Jerry,', 'mouse', 'Jerry'), ('mouse', 'Jerry',
```

```
from ctypes import sizeof
from collections import defaultdict
```

```
# Get NLTK stop words
stop_words = set(stopwords.words("english"))

# Tokenize the text into words
words = corpus_text.split()

# Remove stop words
words = [word for word in words if word.lower() not in stop_words]

# Create a dictionary to store bigram counts
bigram_counts = defaultdict(int)

# Create a dictionary to store conditional counts
conditional_counts = defaultdict(lambda: defaultdict(int))

# Calculate bigram counts
for i in range(len(words) - 1):
    bigram = (words[i], words[i + 1])
    bigram_counts[bigram] += 1

# Calculate conditional counts
for bigram, count in bigram_counts.items():
    conditional_counts[bigram[0]][bigram[1]] = count

# Calculate bigram probabilities
bigram_probabilities = {}
for bigram, count in bigram_counts.items():
    probability = (count / len(conditional_counts))
    bigram_probabilities[bigram] = probability
```

```
df = pd.DataFrame(bigram_counts.items(), columns=["bigram", "count"])
df
```

	bigram	count
0	(cat, Tom)	1
1	(Tom, chased)	1
2	(chased, mouse)	1
3	(mouse, Jerry,)	1
4	(Jerry,, mouse)	1
5	(mouse, Jerry)	1
6	(Jerry, ran)	1
7	(ran, away)	1
8	(away, cat)	1
9	(cat, Tom.)	1

```
bigram_probabilities = [ [ bigram, freq / total_bigrams ] for bigram, freq in bigram_counts.items() ]
df = pd.DataFrame( bigram_probabilities, columns=["bigram", "probability"])
df
```

	bigram	probability
0	(cat, Tom)	0.083333
1	(Tom, chased)	0.083333
2	(chased, mouse)	0.083333
3	(mouse, Jerry,)	0.083333
4	(Jerry,, mouse)	0.083333
5	(mouse, Jerry)	0.083333
6	(Jerry, ran)	0.083333
7	(ran, away)	0.083333
8	(away, cat)	0.083333
9	(cat, Tom.)	0.083333