```
In [2]: import os
        import nltk
        nltk.download()

        showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/inde
        x.xml
Out[2]: True

In [4]: AI="""Certainly! AI, or Artificial Intelligence, refers to the development a
        AI

Out[4]: 'Certainly! AI, or Artificial Intelligence, refers to the development and im
        plementation of computer systems that can perform tasks that typically requi
        re human intelligence. AI aims to simulate human cognitive abilities such as
        learning, problem-solving, perception, and language understanding'

In [5]: type(AI)

Out[5]: str

In [6]: from nltk.tokenize import word_tokenize

In [7]: tokens = word_tokenize(AI)
        tokens
```

```
Out[7]: ['Certainly',
 '!',
 'AI',
 ',',
 'or',
 'Artificial',
 'Intelligence',
 ',',
 'refers',
 'to',
 'the',
 'development',
 'and',
 'implementation',
 'of',
 'computer',
 'systems',
 'that',
 'can',
 'perform',
 'tasks',
 'that',
 'typically',
 'require',
 'human',
 'intelligence',
 '.',
 'AI',
 'aims',
 'to',
 'simulate',
 'human',
 'cognitive',
 'abilities',
 'such',
 'as',
 'learning',
 ',',
 'problem-solving',
 ',',
 'perception',
 ',',
 'and',
 'language',
 'understanding']
```

```
In [8]: len(tokens)
```

```
Out[8]: 45
```

```
In [9]: from nltk.tokenize import sent_tokenize
```

```
In [11]: sentences = sent_tokenize(AI)
         sentences
```

```
Out[11]: ['Certainly!',
 'AI, or Artificial Intelligence, refers to the development and implementati
on of computer systems that can perform tasks that typically require human i
ntelligence.',
 'AI aims to simulate human cognitive abilities such as learning, problem-so
lving, perception, and language understanding']
```

```
In [12]: len(sentences)
```

```
Out[12]: 3
```

```
In [13]: from nltk.tokenize import WhitespaceTokenizer
```

```
In [14]: tokenizer = WhitespaceTokenizer()
```

```
In [16]: tokens1 = tokenizer.tokenize(AI)#no full stop
         tokens1
```

```
Out[16]: ['Certainly!',
          'AI,',
          'or',
          'Artificial',
          'Intelligence,',
          'refers',
          'to',
          'the',
          'development',
          'and',
          'implementation',
          'of',
          'computer',
          'systems',
          'that',
          'can',
          'perform',
          'tasks',
          'that',
          'typically',
          'require',
          'human',
          'intelligence.',
          'AI',
          'aims',
          'to',
          'simulate',
          'human',
          'cognitive',
          'abilities',
          'such',
          'as',
          'learning,',
          'problem-solving,',
          'perception,',
          'and',
          'language',
          'understanding']
```

```
In [17]: from nltk.tokenize import blankline_tokenize#give ypu pargraph
```

```
In [18]: AI_Blank=blankline_tokenize(AI)
         AI_Blank
```

```
Out[18]: ['Certainly! AI, or Artificial Intelligence, refers to the development and i
         mplementation of computer systems that can perform tasks that typically requ
         ire human intelligence. AI aims to simulate human cognitive abilities such a
         s learning, problem-solving, perception, and language understanding']
```

```
In [19]: len(AI_Blank)
```

```
Out[19]: 1
```

In [21]:
```
#you can do same operation in spyder
# 3types of tokeniziation
#Bigram , trigram and ngram
from nltk.util import bigrams, trigrams, ngrams
```

In [22]:
```
# Tokenize your text
string="This is a sample sentence for n-gram tokenization."
tokens = nltk.word_tokenize(string)
tokens
```

Out[22]:
```
['This', 'is', 'a', 'sample', 'sentence', 'for', 'n-gram', 'tokenization',
'.']
```

In [23]:
```
len(tokens)
```

Out[23]:
```
9
```

In [26]:
```
# Generate bigrams, we create two words together anot miss the data
bi_grams = list(nltk.bigrams(tokens))
bi_grams
```

Out[26]:
```
[('This', 'is'),
 ('is', 'a'),
 ('a', 'sample'),
 ('sample', 'sentence'),
 ('sentence', 'for'),
 ('for', 'n-gram'),
 ('n-gram', 'tokenization'),
 ('tokenization', '.')]
```

In [28]:
```
tri_grams = list(nltk.trigrams(tokens))
tri_grams
```

Out[28]:
```
[('This', 'is', 'a'),
 ('is', 'a', 'sample'),
 ('a', 'sample', 'sentence'),
 ('sample', 'sentence', 'for'),
 ('sentence', 'for', 'n-gram'),
 ('for', 'n-gram', 'tokenization'),
 ('n-gram', 'tokenization', '.')]
```

In [29]:
```
#ngrams
four_grams = list(nltk.ngrams(tokens, 4))
four_grams
```

Out[29]:
```
[('This', 'is', 'a', 'sample'),
 ('is', 'a', 'sample', 'sentence'),
 ('a', 'sample', 'sentence', 'for'),
 ('sample', 'sentence', 'for', 'n-gram'),
 ('sentence', 'for', 'n-gram', 'tokenization'),
 ('for', 'n-gram', 'tokenization', '.')]
```

In [30]:
```
five_grams = list(nltk.ngrams(tokens, 5))
five_grams
```

Out[30]:
```
[('This', 'is', 'a', 'sample', 'sentence'),
 ('is', 'a', 'sample', 'sentence', 'for'),
 ('a', 'sample', 'sentence', 'for', 'n-gram'),
 ('sample', 'sentence', 'for', 'n-gram', 'tokenization'),
 ('sentence', 'for', 'n-gram', 'tokenization', '.')]
```

In [31]:
```
len(five_grams)
```

Out[31]:
```
5
```

In [32]:
```
##NLU
#stemming, normalize word into base form
from nltk.stem import PorterStemmer
```

In [33]:
```
stemmer = PorterStemmer()
```

In [35]:
```
word = "running"
stemmed_word = stemmer.stem(word)
stemmed_word
```

Out[35]:
```
'run'
```

In [36]:
```
words = ["running", "played", "eating"]

stemmer = PorterStemmer()

for word in words:
    stemmed_word = stemmer.stem(word)
    print(f"Original word: {word}, Stemmed word: {stemmed_word}")
```

```
Original word: running, Stemmed word: run
Original word: played, Stemmed word: play
Original word: eating, Stemmed word: eat
```

In [37]:
```
words = ["running", "played", "eating","sleep","drink"]
stemmer = PorterStemmer()

for word in words:
    stemmed_word = stemmer.stem(word)
    print(f"Original word: {word}, Stemmed word: {stemmed_word}")
```

```
Original word: running, Stemmed word: run
Original word: played, Stemmed word: play
Original word: eating, Stemmed word: eat
Original word: sleep, Stemmed word: sleep
Original word: drink, Stemmed word: drink
```

In [54]:
```
from nltk.stem import LancasterStemmer

words = ["running","giving","played", "eating"]
lst= LancasterStemmer()

for word in words:
    stemmed_word = lst.stem(word)
    print(word + ": " + stemmed_word)
```

```
running: run
giving: giv
played: play
eating: eat
```

In [62]:
```
from nltk.stem import SnowballStemmer#result like porterstemer give base wor

words = ["running","giving","played", "eating"]
snb= SnowballStemmer("english")

for word in words:
    stemmed_word = snb.stem(word)
    print(word + ": " + stemmed_word)
```

```
running: run
giving: give
played: play
eating: eat
```

In [63]:
```
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
```

In [66]:
```
word_lem=WordNetLemmatizer()
for word in words:
    stemmed_word = word_lem.lemmatize(word)
    print(word + ": " + stemmed_word)
```

```
running: running
giving: giving
played: played
eating: eating
```

In [67]:
```
#stopwrods
from nltk.corpus import stopwords
```

In [68]:
```
stopwords.words("english")
```

```
Out[68]: ['i',
          'me',
          'my',
          'myself',
          'we',
          'our',
          'ours',
          'ourselves',
          'you',
          "you're",
          "you've",
          "you'll",
          "you'd",
          'your',
          'yours',
          'yourself',
          'yourselves',
          'he',
          'him',
          'his',
          'himself',
          'she',
          "she's",
          'her',
          'hers',
          'herself',
          'it',
          "it's",
          'its',
          'itself',
          'they',
          'them',
          'their',
          'theirs',
          'themselves',
          'what',
          'which',
          'who',
          'whom',
          'this',
          'that',
          "that'll",
          'these',
          'those',
          'am',
          'is',
          'are',
          'was',
          'were',
          'be',
          'been',
          'being',
          'have',
          'has',
          'had',
          'having',
          'do',
          'does',
          'did',
          'doing',
          'a',
          'an',
          'the',
          'and',
          'but',
          'if',
          'or',
          'because',
          'as',
          'until',
          'while',
          'of',
          'at',
          'by',
          'for',
          'with',
          'about',
          'against',
          'between',
          'into',
          'through',
          'during',
          'before',
          'after',
          'above',
          'below',
          'to',
          'from',
          'up',
          'down',
          'in',
          'out',
          'on',
          'off',
          'over',
          'under',
          'again',
          'further',
          'then',
          'once',
          'here',
          'there',
          'when',
          'where',
          'why',
          'how',
          'all',
          'any',
          'both',
          'each',
          'few',
          'more',
          'most',
          'other',
          'some',
          'such',
          'no',
          'nor',
          'not',
          'only',
          'own',
          'same',
          'so',
          'than',
          'too',
          'very',
          's',
          't',
          'can',
          'will',
          'just',
          'don',
          "don't",
          'should',
          "should've",
          'now',
          'd',
          'll',
          'm',
          'o',
          're',
          've',
          'y',
          'ain',
          'aren',
          "aren't",
          'couldn',
          "couldn't",
          'didn',
          "didn't",
          'doesn',
          "doesn't",
          'hadn',
          "hadn't",
          'hasn',
          "hasn't",
          'haven',
          "haven't",
          'isn',
          "isn't",
          'ma',
          'mightn',
          "mightn't",
          'mustn',
          "mustn't",
          'needn',
          "needn't",
          'shan',
          "shan't",
          'shouldn',
          "shouldn't",
          'wasn',
          "wasn't",
          'weren',
          "weren't",
          'won',
          "won't",
          'wouldn',
          "wouldn't"]
```

```
In [69]: len(stopwords.words("english"))
```

```
Out[69]: 179
```

```
In [70]: stopwords.words("finnish")
```

```
Out[70]: ['olla',
          'olen',
          'olet',
          'on',
          'olemme',
          'olette',
          'ovat',
          'ole',
          'oli',
          'olisi',
          'olisit',
          'olisin',
          'olisimme',
          'olisitte',
          'olisivat',
          'olit',
          'olin',
          'olimme',
          'olitte',
          'olivat',
          'ollut',
          'olleet',
          'en',
          'et',
          'ei',
          'emme',
          'ette',
          'eivät',
          'minä',
          'minun',
          'minut',
          'minua',
          'minussa',
          'minusta',
          'minuun',
          'minulla',
          'minulta',
          'minulle',
          'sinä',
          'sinun',
          'sinut',
          'sinua',
          'sinussa',
          'sinusta',
          'sinuun',
          'sinulla',
          'sinulta',
          'sinulle',
          'hän',
          'hänen',
          'hänet',
          'häntä',
          'hänessä',
          'hänestä',
          'häneen',
          'hänellä',
          'häneltä',
          'hänelle',
          'me',
          'meidän',
          'meidät',
          'meitä',
          'meissä',
          'meistä',
          'meihin',
          'meillä',
          'meiltä',
          'meille',
          'te',
          'teidän',
          'teidät',
          'teitä',
          'teissä',
          'teistä',
          'teihin',
          'teillä',
          'teiltä',
          'teille',
          'he',
          'heidän',
          'heidät',
          'heitä',
          'heissä',
          'heistä',
          'heihin',
          'heillä',
          'heiltä',
          'heille',
          'tämä',
          'tämän',
          'tätä',
          'tässä',
          'tästä',
          'tähän',
          'tallä',
          'tältä',
          'tälle',
          'tänä',
          'täksi',
          'tuo',
          'tuon',
          'tuotä',
          'tuossa',
          'tuosta',
          'tuohon',
          'tuolla',
          'tuolta',
          'tuolle',
          'tuona',
          'tuoksi',
          'se',
          'sen',
          'sitä',
          'siinä',
          'siitä',
          'siihen',
          'sillä',
          'siltä',
          'sille',
          'sinä',
          'siksi',
          'nämä',
          'näiden',
          'näitä',
          'näissä',
          'näistä',
          'näihin',
          'näillä',
          'näiltä',
          'näille',
          'näinä',
          'näiksi',
          'nuo',
          'noiden',
          'noita',
          'noissa',
          'noista',
          'noihin',
          'noilla',
          'noilta',
          'noille',
          'noina',
          'noiksi',
          'ne',
          'niiden',
          'niitä',
          'niissä',
          'niistä',
          'niihin',
          'niillä',
          'niiltä',
          'niille',
          'niinä',
          'niiksi',
          'kuka',
          'kenen',
          'kenet',
          'ketä',
          'kenessä',
          'kenestä',
          'keneen',
          'kenellä',
          'keneltä',
          'kenelle',
          'kenenä',
          'keneksi',
          'ketkä',
          'keiden',
          'ketkä',
          'keitä',
          'keissä',
          'keistä',
          'keihin',
          'keillä',
          'keiltä',
          'keille',
          'keinä',
          'keiksi',
          'mikä',
          'minkä',
          'minkä',
          'mitä',
          'missä',
          'mistä',
          'mihin',
          'millä',
          'miltä',
          'mille',
          'minä',
          'miksi',
          'mitkä',
          'joka',
          'jonka',
          'jota',
          'jossa',
          'josta',
          'johon',
          'jolla',
          'jolta',
          'jolle',
          'jona',
          'joksi',
          'jotka',
          'joiden',
          'joita',
          'joissa',
          'joista',
          'joihin',
          'joilla',
          'joilta',
          'joille',
          'joina',
          'joiksi',
          'että',
          'ja',
          'jos',
          'koska',
          'kuin',
          'mutta',
          'niin',
          'sekä',
          'sillä',
          'tai',
          'vaan',
          'vai',
          'vaikka',
          'kanssa',
          'mukaan',
          'noin',
          'poikki',
          'yli',
          'kun',
          'niin',
          'nyt',
          'itse']
```

In [71]:
```
len(stopwords.words("finnish"))
```

Out[71]:
```
235
```

In [73]:
```
len(stopwords.words("german"))
```

Out[73]:
```
232
```

In [74]:
```
import nltk
from nltk.stem import PorterStemmer
```

In [75]:
```
from nltk.corpus import stopwords
```

In [79]:
```
ti="""Certainly! AI, or Artificial Intelligence, refers to the development a
ti
```

```
Out[79]: 'Certainly! AI, or Artificial Intelligence, refers to the development and im
plementation of computer systems that can perform tasks that typically requi
re human intelligence. AI aims to simulate human cognitive abilities such as
learning, problem-solving, perception, and language understanding'
```

```
In [80]: sentences = nltk.sent_tokenize(ti)
sentences
```

```
Out[80]: ['Certainly!',
 'AI, or Artificial Intelligence, refers to the development and implementati
on of computer systems that can perform tasks that typically require human i
ntelligence.',
 'AI aims to simulate human cognitive abilities such as learning, problem-so
lving, perception, and language understanding']
```

```
In [81]: len(sentences)
```

```
Out[81]: 3
```

```
In [82]: #import stopwords
# I want to remove all the stopwords from my senterences
# if you check the stopwords.words('english') you get a list of word which i
# you do get stopwords in many language.
# after removing the stopwords i am going to stem the words by using portste

# using for loop for all of sentences & using word_tokenize will convert all
# basically i am writhing for word in words and i am taking from unique word
# Stemming
for i in range(len(sentences)):
    words = nltk.word_tokenize(sentences[i])
    words = [stemmer.stem(word) for word in words if word not in set(stopwor
    sentences[i] = ' '.join(words)
```

```
In [83]: sentences
```

```
Out[83]: ['certain !',
 'ai , art intellig , ref develop impl comput system perform task typ requir
hum intellig .',
 'ai aim sim hum cognit abl learn , problem-solving , perceiv , langu unders
tand']
```

```
In [ ]:
```