Natural Language Processing (NLP)

```
In [1]: import os
        import nltk
```

```
In [2]: import nltk.corpus
```

```
In [3]: # The data can be anything
        AI =  '''Artificial Intelligence refers to the intelligence of machines. This is in
```

```
In [4]: AI
```

```
Out[4]: 'Artificial Intelligence refers to the intelligence of machines. This is in contra
        st to the natural intelligence of humans and animals. With Artificial Intelligenc
        e, machines perform functions such as learning, planning, reasoning and problem-so
        lving. Most noteworthy, Artificial Intelligence is the simulation of human intelli
        gence by machines. It is probably the fastest-growing development in the World of
        technology and innovation. Furthermore, many experts believe AI could solve major
        challenges and crisis situations.'
```

```
In [5]: type(AI)              # String
```

```
Out[5]: str
```

Tokenization

Word Tokenize

```
In [6]: from nltk.tokenize import word_tokenize            # word_tokenize breaksdown the d
```

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

```
In [8]: tokens
```

```
Out[8]: ['Artificial',
         'Intelligence',
         'refers',
         'to',
         'the',
         'intelligence',
         'of',
         'machines',
         '.',
         'This',
         'is',
         'in',
         'contrast',
         'to',
         'the',
         'natural',
         'intelligence',
         'of',
         'humans',
         'and',
         'animals',
         '.',
         'With',
         'Artificial',
         'Intelligence',
         ',',
         'machines',
         'perform',
         'functions',
         'such',
         'as',
         'learning',
         ',',
         'planning',
         ',',
         'reasoning',
         'and',
         'problem-solving',
         '.',
         'Most',
         'noteworthy',
         ',',
         'Artificial',
         'Intelligence',
         'is',
         'the',
         'simulation',
         'of',
         'human',
         'intelligence',
         'by',
         'machines',
         '.',
         'It',
         'is',
         'probably',
 'the',
 'fastest-growing',
 'development',
 'in',
 'the',
 'World',
 'of',
 'technology',
 'and',
 'innovation',
 '.',
 'Furthermore',
 ',',
 'many',
 'experts',
 'believe',
 'AI',
 'could',
 'solve',
 'major',
 'challenges',
 'and',
 'crisis',
 'situations',
 '.']
```

In [9]:
```
len(tokens)
```

Out[9]:
```
81
```

Sentence Tokenization

In [10]:
```
from nltk.tokenize import sent_tokenize            # sent_tokenize breaksdown th
```

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

Out[11]:
```
['Artificial Intelligence refers to the intelligence of machines.',
 'This is in contrast to the natural intelligence of humans and animals.',
 'With Artificial Intelligence, machines perform functions such as learning, plann
ing, reasoning and problem-solving.',
 'Most noteworthy, Artificial Intelligence is the simulation of human intelligence
by machines.',
 'It is probably the fastest-growing development in the World of technology and in
novation.',
 'Furthermore, many experts believe AI could solve major challenges and crisis sit
uations.']
```

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

Out[12]:
```
6
```

Blank Line Tokenize

```
In [13]: from nltk.tokenize import blankline_tokenize               # blankline_tokenize giv
```

```
In [14]: blank_tokens = blankline_tokenize(AI)
```

```
In [15]: blank_tokens
```

```
Out[15]: ['Artificial Intelligence refers to the intelligence of machines. This is in contr
         ast to the natural intelligence of humans and animals. With Artificial Intelligenc
         e, machines perform functions such as learning, planning, reasoning and problem-so
         lving. Most noteworthy, Artificial Intelligence is the simulation of human intelli
         gence by machines. It is probably the fastest-growing development in the World of
         technology and innovation. Furthermore, many experts believe AI could solve major
         challenges and crisis situations.']
```

```
In [16]: len(blank_tokens)
```

```
Out[16]: 1
```

White Space Tokenizer

```
In [17]: from nltk.tokenize import WhitespaceTokenizer            # whitespacetokenizer is si
```

```
In [18]: white_tokens = WhitespaceTokenizer().tokenize(AI)
         white_tokens
```

```
Out[18]: ['Artificial',
          'Intelligence',
          'refers',
          'to',
          'the',
          'intelligence',
          'of',
          'machines.',
          'This',
          'is',
          'in',
          'contrast',
          'to',
          'the',
          'natural',
          'intelligence',
          'of',
          'humans',
          'and',
          'animals.',
          'With',
          'Artificial',
          'Intelligence,',
          'machines',
          'perform',
          'functions',
          'such',
          'as',
          'learning,',
          'planning,',
          'reasoning',
          'and',
          'problem-solving.',
          'Most',
          'noteworthy,',
          'Artificial',
          'Intelligence',
          'is',
          'the',
          'simulation',
          'of',
          'human',
          'intelligence',
          'by',
          'machines.',
          'It',
          'is',
          'probably',
          'the',
          'fastest-growing',
          'development',
          'in',
          'the',
          'World',
          'of',
          'technology',
          'and',
          'innovation.',
          'Furthermore,',
          'many',
          'experts',
          'believe',
          'AI',
          'could',
          'solve',
          'major',
          'challenges',
          'and',
          'crisis',
          'situations.']
```

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

Out[19]:
```
70
```

In [20]:
```
len(tokens)
```

Out[20]:
```
81
```

In [21]:
```
sentence = "Good Apple cost $3.88 in hyderabad. Please buy me two of them. Thanks."
sentence
```

Out[21]:
```
'Good Apple cost $3.88 in hyderabad. Please buy me two of them. Thanks.'
```

Wordpunct Tokenize

In [22]:
```
from nltk.tokenize import wordpunct_tokenize
```

In [23]:
```
wordpunct_tokenize(sentence)
```

Out[23]:
```
['Good',
 'Apple',
 'cost',
 '$',
 '3',
 '.',
 '88',
 'in',
 'hyderabad',
 '.',
 'Please',
 'buy',
 'me',
 'two',
 'of',
 'them',
 '.',
 'Thanks',
 '.']
```

```
In [24]: len(wordpunct_tokenize(sentence))

Out[24]: 19

In [25]: word_tokenize(sentence)

Out[25]: ['Good',
          'Apple',
          'cost',
          '$',
          '3.88',
          'in',
          'hyderabad',
          '.',
          'Please',
          'buy',
          'me',
          'two',
          'of',
          'them',
          '.',
          'Thanks',
          '.']

In [26]: len(word_tokenize(sentence))

Out[26]: 17

In [27]: wordpunct_tokenize(AI)
```

```
Out[27]: ['Artificial',
          'Intelligence',
          'refers',
          'to',
          'the',
          'intelligence',
          'of',
          'machines',
          '.',
          'This',
          'is',
          'in',
          'contrast',
          'to',
          'the',
          'natural',
          'intelligence',
          'of',
          'humans',
          'and',
          'animals',
          '.',
          'With',
          'Artificial',
          'Intelligence',
          ',',
          'machines',
          'perform',
          'functions',
          'such',
          'as',
          'learning',
          ',',
          'planning',
          ',',
          'reasoning',
          'and',
          'problem',
          '-',
          'solving',
          '.',
          'Most',
          'noteworthy',
          ',',
          'Artificial',
          'Intelligence',
          'is',
          'the',
          'simulation',
          'of',
          'human',
          'intelligence',
          'by',
          'machines',
          '.',
          'It',
          'is',
          'probably',
          'the',
          'fastest',
          '-',
          'growing',
          'development',
          'in',
          'the',
          'World',
          'of',
          'technology',
          'and',
          'innovation',
          '.',
          'Furthermore',
          ',',
          'many',
          'experts',
          'believe',
          'AI',
          'could',
          'solve',
          'major',
          'challenges',
          'and',
          'crisis',
          'situations',
          '.']
```

In [28]:
```
len(wordpunct_tokenize(AI))
```

Out[28]:
```
85
```

In [29]:
```
print(word_tokenize(AI))
print(len(word_tokenize(AI)))
```

```
['Artificial', 'Intelligence', 'refers', 'to', 'the', 'intelligence', 'of', 'machine
s', '.', 'This', 'is', 'in', 'contrast', 'to', 'the', 'natural', 'intelligence', 'o
f', 'humans', 'and', 'animals', '.', 'With', 'Artificial', 'Intelligence', ',', 'mac
hines', 'perform', 'functions', 'such', 'as', 'learning', ',', 'planning', ',', 'rea
soning', 'and', 'problem-solving', '.', 'Most', 'noteworthy', ',', 'Artificial', 'In
telligence', 'is', 'the', 'simulation', 'of', 'human', 'intelligence', 'by', 'machin
es', '.', 'It', 'is', 'probably', 'the', 'fastest-growing', 'development', 'in', 'th
e', 'World', 'of', 'technology', 'and', 'innovation', '.', 'Furthermore', ',', 'man
y', 'experts', 'believe', 'AI', 'could', 'solve', 'major', 'challenges', 'and', 'cri
sis', 'situations', '.']
81
```

In [30]:
```
text = """This is the first pargraph.

This is the second paragraph.

This is the third paragraph."""
```

```
In [31]: paragraphs = blankline_tokenize(text)
```

```
In [32]: paragraphs
```

```
Out[32]: ['This is the first pargraph.',
          'This is the second paragraph.',
          'This is the third paragraph.']
```

```
In [33]: len(paragraphs)
```

```
Out[33]: 3
```

```
In [34]: from nltk.util import bigrams,trigrams,ngrams
```

```
In [35]: string = 'the best and most beautifull thing in the world cannot be seen or even to
         string
```

```
Out[35]: 'the best and most beautifull thing in the world cannot be seen or even touched,th
         ey must be felt with heart'
```

```
In [36]: # Unigrams

         tokens = nltk.word_tokenize(string)
         tokens
```

```
Out[36]: ['the',
          'best',
          'and',
          'most',
          'beautifull',
          'thing',
          'in',
          'the',
          'world',
          'can',
          'not',
          'be',
          'seen',
          'or',
          'even',
          'touched',
          ',',
          'they',
          'must',
          'be',
          'felt',
          'with',
          'heart']
```

```
In [37]: len(tokens)
```

```
Out[37]: 23
```

Bigrams

```
In [38]: # bigrams

         bigram_tokens = list(nltk.bigrams(tokens))
         bigram_tokens
```

```
Out[38]: [('the', 'best'),
          ('best', 'and'),
          ('and', 'most'),
          ('most', 'beautifull'),
          ('beautifull', 'thing'),
          ('thing', 'in'),
          ('in', 'the'),
          ('the', 'world'),
          ('world', 'can'),
          ('can', 'not'),
          ('not', 'be'),
          ('be', 'seen'),
          ('seen', 'or'),
          ('or', 'even'),
          ('even', 'touched'),
          ('touched', ','),
          (',', 'they'),
          ('they', 'must'),
          ('must', 'be'),
          ('be', 'felt'),
          ('felt', 'with'),
          ('with', 'heart')]
```

Trigrams

```
In [39]: # trigrams

         trigram_tokens = list(nltk.trigrams(tokens))
         trigram_tokens
```

```
Out[39]: [('the', 'best', 'and'),
          ('best', 'and', 'most'),
          ('and', 'most', 'beautifull'),
          ('most', 'beautifull', 'thing'),
          ('beautifull', 'thing', 'in'),
          ('thing', 'in', 'the'),
          ('in', 'the', 'world'),
          ('the', 'world', 'can'),
          ('world', 'can', 'not'),
          ('can', 'not', 'be'),
          ('not', 'be', 'seen'),
          ('be', 'seen', 'or'),
          ('seen', 'or', 'even'),
          ('or', 'even', 'touched'),
          ('even', 'touched', ','),
          ('touched', ',', 'they'),
          (',', 'they', 'must'),
          ('they', 'must', 'be'),
          ('must', 'be', 'felt'),
          ('be', 'felt', 'with'),
          ('felt', 'with', 'heart')]
```

NGrams

```
In [41]: ngrams_token = list(nltk.ngrams(tokens,5))
         ngrams_token
```

```
Out[41]: [('the', 'best', 'and', 'most', 'beautifull'),
          ('best', 'and', 'most', 'beautifull', 'thing'),
          ('and', 'most', 'beautifull', 'thing', 'in'),
          ('most', 'beautifull', 'thing', 'in', 'the'),
          ('beautifull', 'thing', 'in', 'the', 'world'),
          ('thing', 'in', 'the', 'world', 'can'),
          ('in', 'the', 'world', 'can', 'not'),
          ('the', 'world', 'can', 'not', 'be'),
          ('world', 'can', 'not', 'be', 'seen'),
          ('can', 'not', 'be', 'seen', 'or'),
          ('not', 'be', 'seen', 'or', 'even'),
          ('be', 'seen', 'or', 'even', 'touched'),
          ('seen', 'or', 'even', 'touched', ','),
          ('or', 'even', 'touched', ',', 'they'),
          ('even', 'touched', ',', 'they', 'must'),
          ('touched', ',', 'they', 'must', 'be'),
          (',', 'they', 'must', 'be', 'felt'),
          ('they', 'must', 'be', 'felt', 'with'),
          ('must', 'be', 'felt', 'with', 'heart')]
```

Stemming

Porter Stemmer

In [42]:
```
from nltk.stem import PorterStemmer
pst = PorterStemmer()
```

In [43]:
```
pst.stem("affection")        # suffix is deleted
```

Out[43]:
```
'affect'
```

In [44]:
```
word = ["giving","given","running","jumped","happily","quickly","foxes"]
for words in word:
    print(words+":"+pst.stem(words))
```

```
giving:give
given:given
running:run
jumped:jump
happily:happili
quickly:quickli
foxes:fox
```

In [45]:
```
from nltk.stem import LancasterStemmer                                # LancasterStemmer
lst = LancasterStemmer()
```

In [48]:
```
words_to_stemm = ["giving","given","running","jumped","happily","quickly","foxes"]
for words in words_to_stemm:
    print(words+":"+lst.stem(words))
```

```
giving:giv
given:giv
running:run
jumped:jump
happily:happy
quickly:quick
foxes:fox
```

In [49]:
```
# for snowball stemmer argument language is always necessary
from nltk.stem import SnowballStemmer
sst = SnowballStemmer("english")
```

In [50]:
```
for words in words_to_stemm:
    print(words+":"+sst.stem(words))
```

```
giving:give
given:given
running:run
jumped:jump
happily:happili
quickly:quick
foxes:fox
```

Lemmatization using Tokens

In [51]:
```
from nltk.stem import WordNetLemmatizer
```

In [52]:
```
lemma = WordNetLemmatizer()
from nltk.tokenize import word_tokenize
```

In [53]:
```
text = "The Stripped bats were hanging on the trees and flying around and better."
tokens = word_tokenize(text)
```

In [54]:
```
tokens
```

Out[54]:
```
['The',
 'Stripped',
 'bats',
 'were',
 'hanging',
 'on',
 'the',
 'trees',
 'and',
 'flying',
 'around',
 'and',
 'better',
 '.']
```

In [55]:
```
import nltk
from nltk.stem import wordnet
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
```

In [58]:
```
lemma = WordNetLemmatizer()
text = "The Stripped bats were hanging on the trees and flying around and better."
tokens = word_tokenize(text)
```

In [59]:
```
tokens
```

Out[59]:
```
['The',
 'Stripped',
 'bats',
 'were',
 'hanging',
 'on',
 'the',
 'trees',
 'and',
 'flying',
 'around',
 'and',
 'better',
 '.']
```

In [60]:
```
pos_tags = nltk.pos_tag(tokens)
```

```
In [61]: pos_tags
```

```
Out[61]: [('The', 'DT'),
          ('Stripped', 'NNP'),
          ('bats', 'NNS'),
          ('were', 'VBD'),
          ('hanging', 'VBG'),
          ('on', 'IN'),
          ('the', 'DT'),
          ('trees', 'NNS'),
          ('and', 'CC'),
          ('flying', 'VBG'),
          ('around', 'RB'),
          ('and', 'CC'),
          ('better', 'JJR'),
          ('.', '.')]
```

```
In [62]: def pos(tag):
             if tag.startswith("J"):
                 return wordnet.ADJ
             elif tag.startswith("V"):
                 return wordnet.VERB
             elif tag.startswith("N"):
                 return wordnet.NOUN
             elif tag.startswith("R"):
                 return wordnet.ADV
             else:
                 return wordnet.NOUN
```

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

```
In [64]: lemmatized_words = [lemma.lemmatize(word,pos(tag)) for word,tag in pos_tags]
```

```
In [65]: print("Original Sentence : " ,text)
         print("Lemmatized sentence : "," ".join(lemmatized_words))
```

```
Original Sentence :  The Stripped bats were hanging on the trees and flying around a
nd better.
Lemmatized sentence :  The Stripped bat be hang on the tree and fly around and good
.
```

Stopwords

```
In [66]: from nltk.corpus import stopwords
```

```
In [67]: stopwords.words("english")
```

```
Out[67]: ['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 [68]: len(stopwords.words("english"))               # the stopwords in english language
```

```
Out[68]: 179
```

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

```
Out[69]: ['au',
          'aux',
          'avec',
          'ce',
          'ces',
          'dans',
          'de',
          'des',
          'du',
          'elle',
          'en',
          'et',
          'eux',
          'il',
          'ils',
          'je',
          'la',
          'le',
          'les',
          'leur',
          'lui',
          'ma',
          'mais',
          'me',
          'même',
          'mes',
          'moi',
          'mon',
          'ne',
          'nos',
          'notre',
          'nous',
          'on',
          'ou',
          'par',
          'pas',
          'pour',
          'qu',
          'que',
          'qui',
          'sa',
          'se',
          'ses',
          'son',
          'sur',
          'ta',
          'te',
          'tes',
          'toi',
          'ton',
          'tu',
          'un',
          'une',
          'vos',
          'votre',
          'vous',
          'c',
          'd',
          'j',
          'l',
          'à',
          'm',
          'n',
          's',
          't',
          'y',
          'été',
          'étée',
          'étées',
          'étés',
          'étant',
          'étante',
          'étants',
          'étantes',
          'suis',
          'es',
          'est',
          'sommes',
          'êtes',
          'sont',
          'serai',
          'seras',
          'sera',
          'serons',
          'serez',
          'seront',
          'serais',
          'serait',
          'serions',
          'seriez',
          'seraient',
          'étais',
          'était',
          'étions',
          'étiez',
          'étaient',
          'fus',
          'fut',
          'fûmes',
          'fûtes',
          'furent',
          'sois',
          'soit',
          'soyons',
          'soyez',
          'soient',
          'fusse',
          'fusses',
          'fût',
          'fussions',
          'fussiez',
          'fussent',
'ayant',
'ayante',
'ayantes',
'ayants',
'eu',
'eue',
'eues',
'eus',
'ai',
'as',
'avons',
'avez',
'ont',
'aurai',
'auras',
'aura',
'aurons',
'aurez',
'auront',
'aurais',
'aurait',
'aurions',
'auriez',
'auraient',
'avais',
'avait',
'avions',
'aviez',
'avaient',
'eut',
'eûmes',
'eûtes',
'eurent',
'aie',
'aies',
'ait',
'ayons',
'ayez',
'aient',
'eusse',
'eusses',
'eût',
'eussions',
'eussiez',
'eussent']
```

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

```
Out[70]: 157
```

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

```
Out[71]:  ['aber',
           'alle',
           'allem',
           'allen',
           'aller',
           'alles',
           'als',
           'also',
           'am',
           'an',
           'ander',
           'andere',
           'anderem',
           'anderen',
           'anderer',
           'anderes',
           'anderm',
           'andern',
           'anderr',
           'anders',
           'auch',
           'auf',
           'aus',
           'bei',
           'bin',
           'bis',
           'bist',
           'da',
           'damit',
           'dann',
           'der',
           'den',
           'des',
           'dem',
           'die',
           'das',
           'dass',
           'daß',
           'derselbe',
           'derselben',
           'denselben',
           'desselben',
           'demselben',
           'dieselbe',
           'dieselben',
           'dasselbe',
           'dazu',
           'dein',
           'deine',
           'deinem',
           'deinen',
           'deiner',
           'deines',
           'denn',
           'derer',
           'dessen',
'dich',
'dir',
'du',
'dies',
'diese',
'diesem',
'diesen',
'dieser',
'dieses',
'doch',
'dort',
'durch',
'ein',
'eine',
'einem',
'einen',
'einer',
'eines',
'einig',
'einige',
'einigem',
'einigen',
'einiger',
'einiges',
'einmal',
'er',
'ihn',
'ihm',
'es',
'etwas',
'euer',
'eure',
'eurem',
'euren',
'eurer',
'eures',
'für',
'gegen',
'gewesen',
'hab',
'habe',
'haben',
'hat',
'hatte',
'hatten',
'hier',
'hin',
'hinter',
'ich',
'mich',
'mir',
'ihr',
'ihre',
'ihrem',
'ihren',
'ihrer',
'ihres',
'euch',
'im',
'in',
'indem',
'ins',
'ist',
'jede',
'jedem',
'jeden',
'jeder',
'jedes',
'jene',
'jenem',
'jenen',
'jener',
'jenes',
'jetzt',
'kann',
'kein',
'keine',
'keinem',
'keinen',
'keiner',
'keines',
'können',
'könnte',
'machen',
'man',
'manche',
'manchem',
'manchen',
'mancher',
'manches',
'mein',
'meine',
'meinem',
'meinen',
'meiner',
'meines',
'mit',
'muss',
'musste',
'nach',
'nicht',
'nichts',
'noch',
'nun',
'nur',
'ob',
'oder',
'ohne',
'sehr',
'sein',
'seine',
'seinem',
'seinen',
'seiner',
'seines',
'selbst',
'sich',
'sie',
'ihnen',
'sind',
'so',
'solche',
'solchem',
'solchen',
'solcher',
'solches',
'soll',
'sollte',
'sondern',
'sonst',
'über',
'um',
'und',
'uns',
'unsere',
'unserem',
'unseren',
'unser',
'unseres',
'unter',
'viel',
'vom',
'von',
'vor',
'während',
'war',
'waren',
'warst',
'was',
'weg',
'weil',
'weiter',
'welche',
'welchem',
'welchen',
'welcher',
'welches',
'wenn',
'werde',
'werden',
'wie',
'wieder',
'will',
'wir',
'wird',
'wirst',
'wo',
'wollen',
 'wollte',
 'würde',
 'würden',
 'zu',
 'zum',
 'zur',
 'zwar',
 'zwischen']
```

```
In [72]: len(stopwords.words("german"))
```

```
Out[72]: 232
```

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

```
Out[73]: ['一',
          '一下',
          '一些',
          '一切',
          '一则',
          '一天',
          '一定',
          '一方面',
          '一旦',
          '一时',
          '一来',
          '一样',
          '一次',
          '一片',
          '一直',
          '一致',
          '一般',
          '一起',
          '一边',
          '一面',
          '万一',
          '上下',
          '上升',
          '上去',
          '上来',
          '上述',
          '上面',
          '下列',
          '下去',
          '下来',
          '下面',
          '不一',
          '不久',
          '不仅',
          '不会',
          '不但',
          '不光',
          '不单',
          '不变',
          '不只',
          '不可',
          '不同',
          '不够',
          '不如',
          '不得',
          '不怕',
          '不惟',
          '不成',
          '不拘',
          '不敢',
          '不断',
          '不是',
          '不比',
          '不然',
          '不特',
          '不独',
'不管',
'不能',
'不要',
'不论',
'不足',
'不过',
'不问',
'与',
'与其',
'与否',
'与此同时',
'专门',
'且',
'两者',
'严格',
'严重',
'个',
'个人',
'个别',
'中小',
'中间',
'丰富',
'临',
'为',
'为主',
'为了',
'为什么',
'为什麼',
'为何',
'为着',
'主张',
'主要',
'举行',
'乃',
'乃至',
'么',
'之',
'之一',
'之前',
'之后',
'之後',
'之所以',
'之类',
'乌乎',
'乎',
'乘',
'也',
'也好',
'也是',
'也罢',
'了',
'了解',
'争取',
'于',
'于是',
'于是乎',
'云云',
'互相',
'产生',
'人们',
'人家',
'什么',
'什么样',
'什麽',
'今后',
'今天',
'今年',
'今後',
'仍然',
'从',
'从事',
'从而',
'他',
'他人',
'他们',
'他的',
'代替',
'以',
'以上',
'以下',
'以为',
'以便',
'以免',
'以前',
'以及',
'以后',
'以外',
'以後',
'以来',
'以至',
'以至于',
'以致',
'们',
'任',
'任何',
'任凭',
'任务',
'企图',
'伟大',
'似乎',
'似的',
'但',
'但是',
'何',
'何况',
'何处',
'何时',
'作为',
'你',
'你们',
'你的',
'使得',
'使用',
'例如',
'依',
'依照',
'依靠',
'促进',
'保持',
'俺',
'俺们',
'倘',
'倘使',
'倘或',
'倘然',
'倘若',
'假使',
'假如',
'假若',
'做到',
'像',
'允许',
'充分',
'先后',
'先後',
'先生',
'全部',
'全面',
'兮',
'共同',
'关于',
'其',
'其一',
'其中',
'其二',
'其他',
'其余',
'其它',
'其实',
'其次',
'具体',
'具体地说',
'具体说来',
'具有',
'再者',
'再说',
'冒',
'冲',
'决定',
'况且',
'准备',
'几',
'几乎',
'几时',
'凭',
'凭借',
'出去',
'出来',
'出现',
'分别',
'则',
'别',
'别的',
'别说',
'到',
'前后',
'前者',
'前进',
'前面',
'加之',
'加以',
'加入',
'加强',
'十分',
'即',
'即令',
'即使',
'即便',
'即或',
'即若',
'却不',
'原来',
'又',
'及',
'及其',
'及时',
'及至',
'双方',
'反之',
'反应',
'反映',
'反过来',
'反过来说',
'取得',
'受到',
'变成',
'另',
'另一方面',
'另外',
'只是',
'只有',
'只要',
'只限',
'叫',
'叫做',
'召开',
'叮咚',
'可',
'可以',
'可是',
'可能',
'可见',
'各',
'各个',
'各人',
'各位',
'各地',
'各种',
'各级',
'各自',
'合理',
'同',
'同一',
'同时',
'同样',
'后来',
'后面',
'向',
'向着',
'吓',
'吗',
'否则',
'吧',
'吧哒',
'吱',
'呀',
'呃',
'呕',
'呗',
'呜',
'呜呼',
'呢',
'周围',
'呵',
'呸',
'呼哧',
'咋',
'和',
'咚',
'咦',
'咱',
'咱们',
'咳',
'哇',
'哈',
'哈哈',
'哉',
'哎',
'哎呀',
'哎哟',
'哗',
'哟',
'哦',
'哩',
'哪',
'哪个',
'哪些',
'哪儿',
'哪天',
'哪年',
'哪怕',
'哪样',
'哪边',
'哪里',
'哼',
'哼唷',
'唉',
'啊',
'啐',
'啥',
'啦',
'啪达',
'喂',
'喏',
'喔唷',
'嗡嗡',
'嗬',
'嗯',
'嗳',
'嘎',
'嘎登',
'嘘',
'嘛',
'嘻',
'嘿',
'因',
'因为',
'因此',
'因而',
'固然',
'在',
'在下',
'地',
'坚决',
'坚持',
'基本',
'处理',
'复杂',
'多',
'多少',
'多数',
'多次',
'大力',
'大多数',
'大大',
'大家',
'大批',
'大约',
'大量',
'失去',
'她',
'她们',
'她的',
'好的',
'好象',
'如',
'如上所述',
'如下',
'如何',
'如其',
'如果',
'如此',
'如若',
'存在',
'宁',
'宁可',
'宁愿',
'宁肯',
'它',
'它们',
'它们的',
'它的',
'安全',
'完全',
'完成',
'实现',
'实际',
'宣布',
'容易',
'密切',
'对',
'对于',
'对应',
'将',
'少数',
'尔后',
'尚且',
'尤其',
'就',
'就是',
'就是说',
'尽',
'尽管',
'属于',
'岂但',
'左右',
'巨大',
'巩固',
'己',
'已经',
'帮助',
'常常',
'并',
'并不',
'并不是',
'并且',
'并没有',
'广大',
'广泛',
'应当',
'应用',
'应该',
'开外',
'开始',
'开展',
'引起',
'强烈',
'强调',
'归',
'当',
'当前',
'当时',
'当然',
'当着',
'形成',
'彻底',
'彼',
'彼此',
'往',
'往往',
'待',
'後来',
'後面',
'得',
'得出',
'得到',
'心里',
'必然',
'必要',
'必须',
'怎',
'怎么',
'怎么办',
'怎么样',
'怎样',
'怎麽',
'总之',
'总是',
'总的来看',
'总的来说',
'总的说来',
'总结',
'总而言之',
'恰恰相反',
'您',
'意思',
'愿意',
'慢说',
'成为',
'我',
'我们',
'我的',
'或',
'或是',
'或者',
'战斗',
'所',
'所以',
    '所有',
    '所谓',
    '打',
    '扩大',
    '把',
    '抑或',
    '拿',
    '按',
    '按照',
    '换句话说',
    '换言之',
    '据',
    '掌握',
    '接着',
    '接著',
    '故',
    '故此',
    '整个',
    '方便',
    '方面',
    '旁人',
    '无宁',
    '无法',
    '无论',
    '既',
    '既是',
    '既然',
    '时候',
    '明显',
    '明确',
    '是',
    '是否',
    '是的',
    '显然',
    '显著',
    '普通',
    '普遍',
    '更加',
    '曾经',
    '替',
    '最后',
    '最大',
    '最好',
    '最後',
    '最近',
    '最高',
    '有',
    '有些',
    '有关',
    '有利',
    '有力',
    '有所',
    '有效',
    '有时',
    '有点',
    '有的',
'有着',
'有著',
'望',
'朝',
'朝着',
'本',
'本着',
'来',
'来着',
'极了',
'构成',
'果然',
'果真',
'某',
'某个',
'某些',
'根据',
'根本',
'欢迎',
'正在',
'正如',
'正常',
'此',
'此外',
'此时',
'此间',
'毋宁',
'每',
'每个',
'每天',
'每年',
'每当',
'比',
'比如',
'比方',
'比较',
'毫不',
'没有',
'沿',
'沿着',
'注意',
'深入',
'清楚',
'满足',
'漫说',
'焉',
'然则',
'然后',
'然後',
'然而',
'照',
'照着',
'特别是',
'特殊',
'特点',
'现代',
'现在',
'甚么',
'甚而',
'甚至',
'用',
'由',
'由于',
'由此可见',
'的',
'的话',
'目前',
'直到',
'直接',
'相似',
'相信',
'相反',
'相同',
'相对',
'相对而言',
'相应',
'相当',
'相等',
'省得',
'看出',
'看到',
'看来',
'看看',
'看见',
'真是',
'真正',
'着',
'着呢',
'矣',
'知道',
'确定',
'离',
'积极',
'移动',
'突出',
'突然',
'立即',
'第',
'等',
'等等',
'管',
'紧接着',
'纵',
'纵令',
'纵使',
'纵然',
'练习',
'组成',
'经',
'经常',
'经过',
'结合',
'结果',
'给',
'绝对',
'继续',
'继而',
'维持',
'综上所述',
'罢了',
'考虑',
'者',
'而',
'而且',
'而况',
'而外',
'而已',
'而是',
'而言',
'联系',
'能',
'能否',
'能够',
'腾',
'自',
'自个儿',
'自从',
'自各儿',
'自家',
'自己',
'自身',
'至',
'至于',
'良好',
'若',
'若是',
'若非',
'范围',
'莫若',
'获得',
'虽',
'虽则',
'虽然',
'虽说',
'行为',
'行动',
'表明',
'表示',
'被',
'要',
'要不',
'要不是',
'要不然',
'要么',
'要是',
'要求',
'规定',
'觉得',
'认为',
'认真',
'认识',
'让',
'许多',
'论',
'设使',
'设若',
'该',
'说明',
'诸位',
'谁',
'谁知',
'赶',
'起',
'起来',
'起见',
'趁',
'趁着',
'越是',
'跟',
'转动',
'转变',
'转贴',
'较',
'较之',
'边',
'达到',
'迅速',
'过',
'过去',
'过来',
'运用',
'还是',
'还有',
'这',
'这个',
'这么',
'这么些',
'这么样',
'这么点儿',
'这些',
'这会儿',
'这儿',
'这就是说',
'这时',
'这样',
'这点',
'这种',
'这边',
'这里',
'这麽',
'进入',
'进步',
'进而',
'进行',
'连',
'连同',
'适应',
'适当',
'适用',
'逐步',
'逐渐',
'通常',
'通过',
'造成',
'遇到',
'遭到',
'避免',
'那',
'那个',
'那么',
'那么些',
'那么样',
'那些',
'那会儿',
'那儿',
'那时',
'那样',
'那边',
'那里',
'那麽',
'部分',
'鄙人',
'采取',
'里面',
'重大',
'重新',
'重要',
'鉴于',
'问题',
'防止',
'阿',
'附近',
'限制',
'除',
'除了',
'除此之外',
'除非',
'随',
'随着',
'随著',
'集中',
'需要',
'非但',
'非常',
'非徒',
'靠',
'顺',
'顺着',
'首先',
 '高兴',
 '是不是']
```

```
In [74]: len(stopwords.words("chinese"))
```

```
Out[74]: 841
```

```
In [75]: stopwords.words("hindi")
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
Cell In[75], line 1
----> 1 stopwords.words("hindi")

File ~\anaconda3\Lib\site-packages\nltk\corpus\reader\wordlist.py:21, in WordListCor
pusReader.words(self, fileids, ignore_lines_startswith)
     18 def words(self, fileids=None, ignore_lines_startswith="\n"):
     19     return [
     20         line
---> 21         for line in line_tokenize(self.raw(fileids))
     22         if not line.startswith(ignore_lines_startswith)
     23     ]

File ~\anaconda3\Lib\site-packages\nltk\corpus\reader\api.py:218, in CorpusReader.ra
w(self, fileids)
    216 contents = []
    217 for f in fileids:
--> 218     with self.open(f) as fp:
    219         contents.append(fp.read())
    220 return concat(contents)

File ~\anaconda3\Lib\site-packages\nltk\corpus\reader\api.py:231, in CorpusReader.op
en(self, file)
    223 """
    224 Return an open stream that can be used to read the given file.
    225 If the file's encoding is not None, then the stream will
   (...)
    228 :param file: The file identifier of the file to read.
    229 """
    230 encoding = self.encoding(file)
--> 231 stream = self._root.join(file).open(encoding)
    232 return stream

File ~\anaconda3\Lib\site-packages\nltk\data.py:334, in FileSystemPathPointer.join(s
elf, fileid)
    332 def join(self, fileid):
    333     _path = os.path.join(self._path, fileid)
--> 334     return FileSystemPathPointer(_path)

File ~\anaconda3\Lib\site-packages\nltk\compat.py:41, in py3_data.<locals>._decorato
r(*args, **kwargs)
     39 def _decorator(*args, **kwargs):
     40     args = (args[0], add_py3_data(args[1])) + args[2:]
---> 41     return init_func(*args, **kwargs)

File ~\anaconda3\Lib\site-packages\nltk\data.py:312, in FileSystemPathPointer.__init
__(self, _path)
    310 _path = os.path.abspath(_path)
    311 if not os.path.exists(_path):
--> 312     raise OSError("No such file or directory: %r" % _path)
    313 self._path = _path

OSError: No such file or directory: 'C:\\Users\\rgukt\\AppData\\Roaming\\nltk_data
\\corpora\\stopwords\\hindi'
```

Parts of Speech (POS)

```
In [76]: text = "sam is natural when it comes to drawing"
```

```
In [77]: text
```

```
Out[77]: 'sam is natural when it comes to drawing'
```

```
In [78]: tokens = word_tokenize(text)
         tokens
```

```
Out[78]: ['sam', 'is', 'natural', 'when', 'it', 'comes', 'to', 'drawing']
```

```
In [79]: # we use NLTK library to work with POS
         for token in tokens:
             print(nltk.pos_tag([token]))
```

```
[('sam', 'NN')]
[('is', 'VBZ')]
[('natural', 'JJ')]
[('when', 'WRB')]
[('it', 'PRP')]
[('comes', 'VBZ')]
[('to', 'TO')]
[('drawing', 'VBG')]
```

```
In [80]: # All the tags has a specific description associate with it
```

```
In [81]: text2 = "John is eating a delicious cake"
         text2
```

```
Out[81]: 'John is eating a delicious cake'
```

```
In [82]: tokens = word_tokenize(text2)
         tokens
```

```
Out[82]: ['John', 'is', 'eating', 'a', 'delicious', 'cake']
```

```
In [83]: for token in tokens:
             print(nltk.pos_tag([token]))
```

```
[('John', 'NNP')]
[('is', 'VBZ')]
[('eating', 'VBG')]
[('a', 'DT')]
[('delicious', 'JJ')]
[('cake', 'NN')]
```

Named Entity Recognition

In [84]:
```
from nltk import ne_chunk
```

In [87]:
```
text = "The US president stays in the WHITEHOUSE"
text
```

Out[87]:
```
'The US president stays in the WHITEHOUSE'
```

In [88]:
```
tokens = word_tokenize(text)
```

In [89]:
```
tokens
```

Out[89]:
```
['The', 'US', 'president', 'stays', 'in', 'the', 'WHITEHOUSE']
```

In [90]:
```
pos_tags = nltk.pos_tag(tokens)
pos_tags
```

Out[90]:
```
[('The', 'DT'),
 ('US', 'NNP'),
 ('president', 'NN'),
 ('stays', 'NNS'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('WHITEHOUSE', 'NNP')]
```

In [91]:
```
NER = ne_chunk(pos_tags)           # we are pssing the pos_tags into ne_chunk funct
print(NER)
```

```
(S
  The/DT
  (GSP US/NNP)
  president/NN
  stays/NNS
  in/IN
  the/DT
  (ORGANIZATION WHITEHOUSE/NNP))
```

WordCloud

In [108…
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
```

In [125…
```
words = "The Stripped bat be hang on the tree and fly around and good The sun be sh
```

In [126…
```
wordcloud = WordCloud(width=420,height=200,background_color="black").generate(words
```

In [127…
```
plt.figure(figsize=(5,5))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

energy
bright
flower
cast
light
full
brightly
across
ray
sunlight
tree
shin
sway
fly
Stripped
gently
day
promise
meadow
bat
good
catch
blue
hang
world
breeze
sun
sky
around
green
petal

In [137…
```
img = Image.open('./imagge.jpg')
img
```

Out[137…

In [140…
```
wordcloud = WordCloud(width=420,height=200,background_color="white",scale=10.0,colo
```

In [141…
```
plt.figure(figsize=(5,5))
plt.imshow(wordcloud,interpolation="quadric")
plt.axis("off")
plt.show()
```

sky
breeze
shin
ray
catch
sun
flower
light
blue
bright
around
across
bat
gently
hang
Stripped
petal
promise
full
sunlight
brightly
world
tree
sway
green
good
day
fly
energy
meadow
cast

In [ ]: