Artificial Intellegence:

Artificial Intellegence (AI) refers to simulation of Human Intelligence processes by machines,especially computer system. This process include learning (the ability acquire to and apply knowledge), and reasoning (ability solve problems) and self correction. AI is board field that encompases various subfields, including machine learning,natural language processing,robotics, and computer vision.

How machine understands languages :

-- When Data is text or image or video ,sensor,standing camera satelite -- All these are unstructered data to be handled by AI

Natural Language Processing(NLP):

-- NLP is a technique to deal with when data is text used in AI. we can also say it is intepreter between machines and human language.

Process Text :

-- The NLP frameworks like NLTK,SPACY,STANDFORD NLP,GENSIM to process the text to be machine understandable.

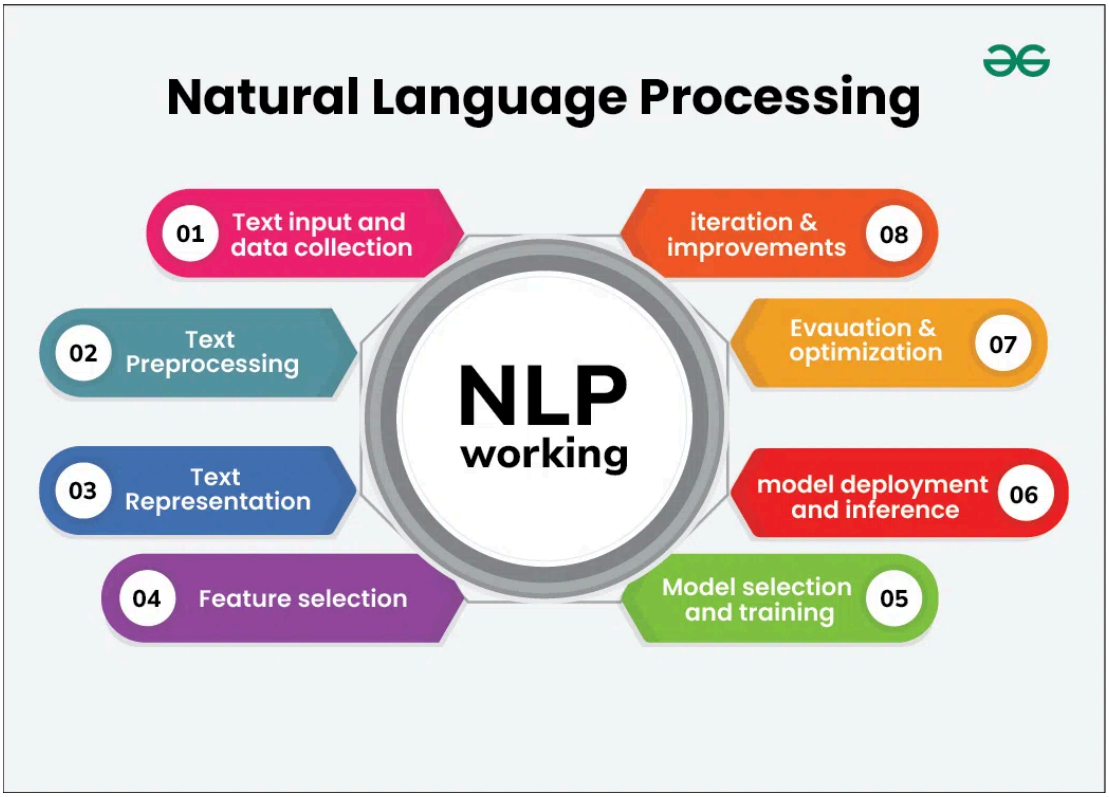
Sample Applications:

Sentimenatal Analysis. ChatBot. Speech Recognition... etc.

Basically NLP is 2 Parts:

-- NLU (Natural Language Understanding) -- NLG (Natural Language Generation)

Natural Language Processing(NLP):



-- NLP is a technique to deal with when data is text used in AI. we can also say it is intepreter between machines and human language.

Process Text :

-- The NLP frameworks like NLTK,SPACY,STANDFORD NLP,GENSIM to process the text to be machine understandable.

Sample Applications:

Sentimenatal Analysis. ChatBot. Speech Recognition... etc.

Basically NLP is 2 Parts:

-- NLU (Natural Language Understanding) -- NLG (Natural Language Generation)

Tokenization:

Tokenization is nothing break a complex sentence into words. The words in nlp is called as tokens.

Word_Tokenize :

The process of splitting text into words or tokens. it treats white space and comma(,),stop(.) also as tokens.

Sent_Tokenize :

Sentenece tokenization is process of break the large set text into small or invididual sentences.

In [1]:
```
# import library to process text data
import os
import nltk
import nltk.corpus
```

In [3]:
```
# Let's take a string try to break it as tokens
AI ='''Artificial Intelligence refers to the intelligence of machines. This is i
humans and animals. With Artificial Intelligence, machines perform functions suc
problem-solving. Most noteworthy, Artificial Intelligence is the simulation of h
It is probably the fastest-growing development in the World of technology and in
AI could solve major challenges and crisis situations.'''
```

In [4]:
```
from nltk.tokenize import word_tokenize
word_token = word_tokenize(AI)
word_token
```

Out[4]:
```
['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 [5]:
```
# check type of AI
type(AI)
```

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

In [6]:
```
len(word_token)
```

Out[6]:
```
81
```

In [7]:
```
from nltk.tokenize import sent_tokenize
```

In [8]:
```
# sent_tokenize will break the large text data into small individual sentences.
AI_sent = sent_tokenize(AI)
AI_sent
```

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

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

Out[9]:
```
6
```

Blankline_tokenize

-- How many paragraphs in input text

In [11]:
```
# blankline_tokenize
from nltk.tokenize import blankline_tokenize

AI_blt = blankline_tokenize(AI) # gives us how many paragraphs in our input
AI_blt
```

Out[11]:
```
['Artificial Intelligence refers to the intelligence of machines. This is in co
ntrast to the natural intelligence of\nhumans and animals. With Artificial Inte
lligence, machines perform functions such as learning, planning, reasoning and
\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation o
f human intelligence by machines.\nIt is probably the fastest-growing developme
nt in the World of technology and innovation. Furthermore, many experts believe
\nAI could solve major challenges and crisis situations.']
```

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

Out[12]:
```
1
```

Whitespace_Tokenizer

--he text is split wherever there is a whitespace character.

In [16]:
```
# whitespace_tokenizer
# The text is split wherever there is a whitespace character.
# It doesn't account for punctuation, which means punctuation marks will be cons
# when we compared to word_tokenizer with whitespace the whitespace will not tre

from nltk.tokenize import WhitespaceTokenizer
wt = WhitespaceTokenizer().tokenize(AI)
wt
```

Out[16]:
```
['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 [15]:
```
len(wt)
```

Out[15]:
```
70
```

Wordprunct_Tokenize

-- we did not call numbers or any special chars to be as tokens in such cases to allow them treat as tokens we need use wordpunct_tokens

In [17]:
```
s ="good apple cost is $3.88 in hyderabad,please buy two of them.Thanks"
s
```

Out[17]:
```
'good apple cost is $3.88 in hyderabad,please buy two of them.Thanks'
```

In [20]:
```
#wordpunct_tokenize
# If we observe there is apple cost $3.88 ,
#usually we did not call numbers or any special chars to be as tokens in such ca
from nltk.tokenize import wordpunct_tokenize
wpt = wordpunct_tokenize(s)
wpt
```

Out[20]:
```
['good',
 'apple',
 'cost',
 'is',
 '$',
 '3',
 '.',
 '88',
 'in',
 'hyderabad',
 ',',
 'please',
 'buy',
 'two',
 'of',
 'them',
 '.',
 'Thanks']
```

In [22]:
```
# if we see AI tokens has incresed with wordpunct_tokenize.
w_p = wordpunct_tokenize(AI)
w_p
len(w_p)
```

Out[22]: 85

Type Of Tokenizations:

1.Bigram

-- tokens with two consecutive words.

2.Trigram

-- tokens with three consecutive words.

3.Ngram

-- tokens with more three consecutive words.

In [23]:
```
from nltk.util import bigrams,trigrams,ngrams
```

In [24]:
```
string = "hello the best and most beatuiful thing in the world can not be seen o
quote_tokens = word_tokenize(string)
quote_tokens
```

Out[24]:
```
['hello',
 'the',
 'best',
 'and',
 'most',
 'beatuiful',
 'thing',
 'in',
 'the',
 'world',
 'can',
 'not',
 'be',
 'seen',
 'or',
 'even',
 'touched',
 ',',
 'they',
 'must',
 'be',
 'felt',
 'with',
 'heart']
```

In [25]:
```
len(quote_tokens)
```

Out[25]: 24

In [26]:
```
string
```

Out[26]:
```
'hello the best and most beatuiful thing in the world can not be seen or even t
ouched,they must be felt with heart'
```

In [27]:
```
quote_tokens
```

Out[27]:
```
['hello',
 'the',
 'best',
 'and',
 'most',
 'beatuiful',
 'thing',
 'in',
 'the',
 'world',
 'can',
 'not',
 'be',
 'seen',
 'or',
 'even',
 'touched',
 ',',
 'they',
 'must',
 'be',
 'felt',
 'with',
 'heart']
```

In [28]:
```
quote_bigrams = list(nltk.bigrams(quote_tokens))
quote_bigrams
```

Out[28]:
```
[('hello', 'the'),
 ('the', 'best'),
 ('best', 'and'),
 ('and', 'most'),
 ('most', 'beatuiful'),
 ('beatuiful', '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')]
```

In [29]:
```
quote_trigrams = list(nltk.trigrams(quote_tokens))
quote_trigrams
```

Out[29]:
```
[('hello', 'the', 'best'),
 ('the', 'best', 'and'),
 ('best', 'and', 'most'),
 ('and', 'most', 'beatuiful'),
 ('most', 'beatuiful', 'thing'),
 ('beatuiful', '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')]
```

In [32]:
```
quote_ngrams = list(nltk.ngrams(quote_tokens,4))
quote_ngrams
```

Out[32]:
```
[('hello', 'the', 'best', 'and'),
 ('the', 'best', 'and', 'most'),
 ('best', 'and', 'most', 'beatuiful'),
 ('and', 'most', 'beatuiful', 'thing'),
 ('most', 'beatuiful', 'thing', 'in'),
 ('beatuiful', 'thing', 'in', 'the'),
 ('thing', 'in', 'the', 'world'),
 ('in', 'the', 'world', 'can'),
 ('the', 'world', 'can', 'not'),
 ('world', 'can', 'not', 'be'),
 ('can', 'not', 'be', 'seen'),
 ('not', 'be', 'seen', 'or'),
 ('be', 'seen', 'or', 'even'),
 ('seen', 'or', 'even', 'touched'),
 ('or', 'even', 'touched', ','),
 ('even', 'touched', ',', 'they'),
 ('touched', ',', 'they', 'must'),
 (',', 'they', 'must', 'be'),
 ('they', 'must', 'be', 'felt'),
 ('must', 'be', 'felt', 'with'),
 ('be', 'felt', 'with', 'heart')]
```

In [33]:
```
quote_ngrams = list(nltk.ngrams(quote_tokens,5))
quote_ngrams
```

Out[33]:
```
[('hello', 'the', 'best', 'and', 'most'),
 ('the', 'best', 'and', 'most', 'beatuiful'),
 ('best', 'and', 'most', 'beatuiful', 'thing'),
 ('and', 'most', 'beatuiful', 'thing', 'in'),
 ('most', 'beatuiful', 'thing', 'in', 'the'),
 ('beatuiful', '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')]
```

In [34]:
```
quote_ngrams = list(nltk.ngrams(quote_tokens,7))
quote_ngrams
```

Out[34]:
```
[('hello', 'the', 'best', 'and', 'most', 'beatuiful', 'thing'),
 ('the', 'best', 'and', 'most', 'beatuiful', 'thing', 'in'),
 ('best', 'and', 'most', 'beatuiful', 'thing', 'in', 'the'),
 ('and', 'most', 'beatuiful', 'thing', 'in', 'the', 'world'),
 ('most', 'beatuiful', 'thing', 'in', 'the', 'world', 'can'),
 ('beatuiful', 'thing', 'in', 'the', 'world', 'can', 'not'),
 ('thing', 'in', 'the', 'world', 'can', 'not', 'be'),
 ('in', 'the', 'world', 'can', 'not', 'be', 'seen'),
 ('the', 'world', 'can', 'not', 'be', 'seen', 'or'),
 ('world', 'can', 'not', 'be', 'seen', 'or', 'even'),
 ('can', 'not', 'be', 'seen', 'or', 'even', 'touched'),
 ('not', 'be', 'seen', 'or', 'even', 'touched', ','),
 ('be', 'seen', 'or', 'even', 'touched', ',', 'they'),
 ('seen', 'or', 'even', 'touched', ',', 'they', 'must'),
 ('or', 'even', 'touched', ',', 'they', 'must', 'be'),
 ('even', 'touched', ',', 'they', 'must', 'be', 'felt'),
 ('touched', ',', 'they', 'must', 'be', 'felt', 'with'),
 (',', 'they', 'must', 'be', 'felt', 'with', 'heart')]
```

In [36]:
```
quote_ngrams = list(nltk.ngrams(quote_tokens,9))
quote_ngrams
```

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

Stemming:

---Normalises the words into root form or base form

In [37]:
```
from nltk.stem import PorterStemmer
pst = PorterStemmer() #
```

In [38]:
```
pst.stem('affection')
```

Out[38]:
```
'affect'
```

In [39]:
```
pst.stem('pefection')
```

Out[39]:
```
'pefect'
```

In [40]:
```
pst.stem('playing')
```

Out[40]: 'play'

In [41]:
```
pst.stem('maximum')
```

Out[41]: 'maximum'

In [42]:
```
words_to_stem = ['give','giving','given','gave']

for words in words_to_stem:
    print(words+":" +pst.stem(words))
```

```
give:give
giving:give
given:given
gave:gave
```

In [43]:
```
word_to_stem =['give','giving','given','gaved','thinking','loving','maximum']
for words in word_to_stem:
     print(words+":" +pst.stem(words))
```

```
give:give
giving:give
given:given
gaved:gave
thinking:think
loving:love
maximum:maximum
```

In [47]:
```
# lets apply lancasterstemmer to words
# lancasterStemmer gives us core root form words
from nltk.stem import LancasterStemmer
lst = LancasterStemmer()

for words in word_to_stem:
     print(words+":" +lst.stem(words))
```

```
give:giv
giving:giv
given:giv
gaved:gav
thinking:think
loving:lov
maximum:maxim
```

In [49]:
```
# for which language we need to stem to be mentioned for snowball stemmer'
# snowball stemmer act like as porter stemmer
from nltk.stem import SnowballStemmer
sbst = SnowballStemmer('english')

for words in word_to_stem:
    print(words+':'+sbst.stem(words))
```

```
give:give
giving:give
given:given
gaved:gave
thinking:think
loving:love
maximum:maximum
```

In [54]:
```
# lemmatization gives a proper word there no removal of suffix and preffix
```

In [50]:
```
from nltk.stem import wordnet
from nltk.stem import WordNetLemmatizer
word_lem = WordNetLemmatizer()
```

In [52]:
```
word_to_stem
```

Out[52]:
```
['give', 'giving', 'given', 'gaved', 'thinking', 'loving', 'maximum']
```

In [53]:
```
for words in word_to_stem:
    print(words+ ':' +word_lem.lemmatize(words))
```

```
give:give
giving:giving
given:given
gaved:gaved
thinking:thinking
loving:loving
maximum:maximum
```

In [55]:
```
from nltk.corpus import stopwords
```

In [56]:
```
stopwords.words('english')
```

Out[56]:
```
['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 [57]:
```
len(stopwords.words('english'))
```

Out[57]: 179

In [58]:
```
stopwords.words('french')
```

Out[58]:
```
['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 [59]:
```
len(stopwords.words('french'))
```

Out[59]: 157

In [60]:
```
stopwords.words('german')
```

Out[60]:
```
['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 [61]:
```
len(stopwords.words('german'))
```

Out[61]: 232

In [62]:
```
stopwords.words('chinese')
```

Out[62]:
```
['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',
'为什麽',
'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 [63]:
```
len(stopwords.words('chinese'))
```

Out[63]: 841

In [65]:
```
stopwords.words('telugu')
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
Cell In[65], line 1
----> 1 stopwords.words('telugu')

File ~\anaconda3\Lib\site-packages\nltk\corpus\reader\wordlist.py:21, in WordList
CorpusReader.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 CorpusReade
r.raw(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 CorpusReade
r.open(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:333, in FileSystemPathPointer.joi
n(self, fileid)
    331 def join(self, fileid):
    332     _path = os.path.join(self._path, fileid)
--> 333     return FileSystemPathPointer(_path)

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

OSError: No such file or directory: 'C:\\Users\\yamini\\AppData\\Roaming\\nltk_da
ta\\corpora\\stopwords\\telugu'
```

Parts Of SPeech

In [66]:
```
sent = 'ram is a natural when it comes to drawing'
sent_tokens = word_tokenize(sent)
sent_tokens
```

Out[66]:
```
['ram', 'is', 'a', 'natural', 'when', 'it', 'comes', 'to', 'drawing']
```

In [68]:
```
for token in sent_tokens:
    print(nltk.pos_tag([token]))
```

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

In [69]:
```
sent2 = 'jhon is eating a delicious cake'
sent2_token = word_tokenize(sent2)

for token in sent2_token:
    print(nltk.pos_tag([token]))
```

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

NER : Named Entity Recognition

In [70]:
```
from nltk import ne_chunk
```

In [71]:
```
NE_sent = 'The US president stays in the WHITEHOUSE'
```

In [72]:
```
NE_tokens = word_tokenize(NE_sent)
NE_tokens
```

Out[72]:
```
['The', 'US', 'president', 'stays', 'in', 'the', 'WHITEHOUSE']
```

In [74]:
```
NE_tags = nltk.pos_tag(NE_tokens)
NE_tags
```

Out[74]:
```
[('The', 'DT'),
 ('US', 'NNP'),
 ('president', 'NN'),
 ('stays', 'NNS'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('WHITEHOUSE', 'NNP')]
```

In [75]:
```
NE_NER = ne_chunk(NE_tags)
print(NE_NER)
```

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

Natural Language Generation

To visualize the text data we need install wordCloud

In [83]:
```
!pip install WordCloud
```

```
Requirement already satisfied: WordCloud in c:\users\yamini\anaconda3\lib\site-pa
ckages (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in c:\users\yamini\anaconda3\lib\site
-packages (from WordCloud) (1.26.4)
Requirement already satisfied: pillow in c:\users\yamini\anaconda3\lib\site-packa
ges (from WordCloud) (11.0.0)
Requirement already satisfied: matplotlib in c:\users\yamini\anaconda3\lib\site-p
ackages (from WordCloud) (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\yamini\anaconda3\lib
\site-packages (from matplotlib->WordCloud) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\yamini\anaconda3\lib\site
-packages (from matplotlib->WordCloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\yamini\anaconda3\lib
\site-packages (from matplotlib->WordCloud) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\yamini\anaconda3\lib
\site-packages (from matplotlib->WordCloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\yamini\anaconda3\lib\s
ite-packages (from matplotlib->WordCloud) (24.1)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\yamini\anaconda3\lib
\site-packages (from matplotlib->WordCloud) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\yamini\anaconda3
\lib\site-packages (from matplotlib->WordCloud) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\yamini\anaconda3\lib\site-pac
kages (from python-dateutil>=2.7->matplotlib->WordCloud) (1.16.0)
```

In [84]:
```
# Libraries
from wordcloud import  WordCloud
import matplotlib.pyplot as plt
```

In [78]:
```
text=("Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Ch
```

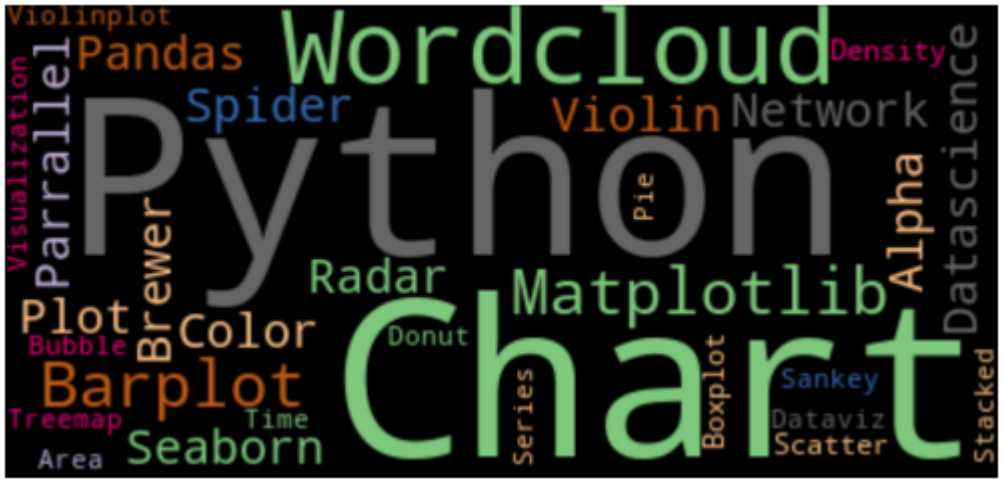
In [80]:
```
text # frequency of name repeats multiple times
```

Out[80]:
```
'Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Chart P
andas Datascience Wordcloud Spider Radar Parrallel Alpha Color Brewer Density S
catter Barplot Barplot Boxplot Violinplot Treemap Stacked Area Chart Chart Visu
alization Dataviz Donut Pie Time-Series Wordcloud Wordcloud Sankey Bubble'
```

In [85]:
```
word_cloud = WordCloud(width=420,height=200,background_color='black',mode='RGBA'
```

In [86]:
```
# disply the generated image
plt.imshow(word_cloud,interpolation='quadric')
```

```
plt.axis('off')
plt.margins(x=0,y=0)
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: