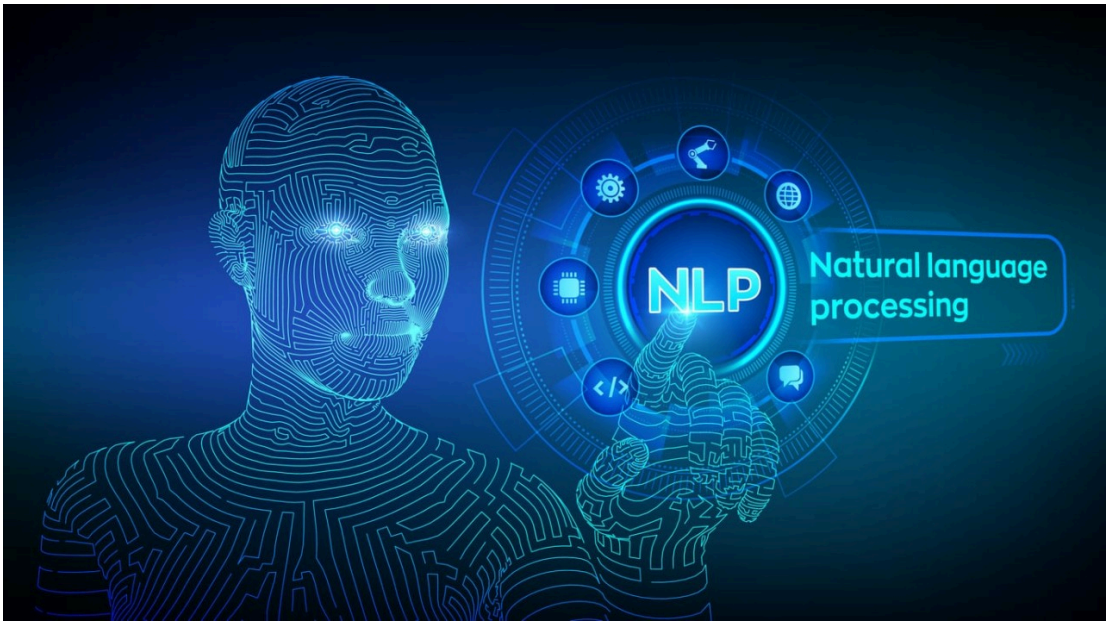
NATURAL LANGUAGE PROCESSING

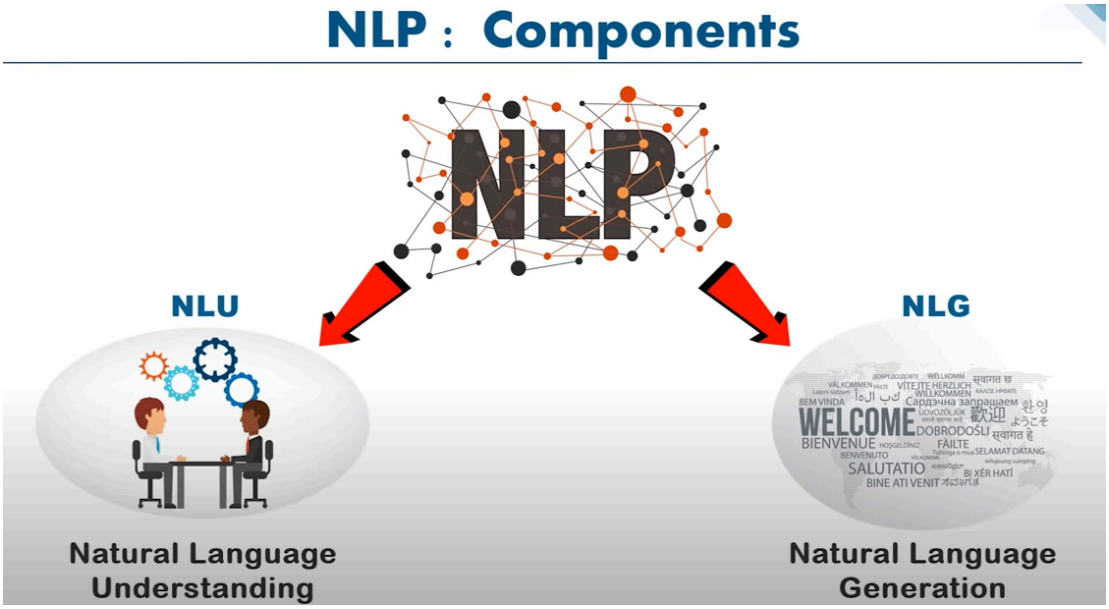
In [2]:
```
import os
import nltk
#nltk.download()
```



Natural language processing (NLP) is a branch of artificial intelligence (AI) that allows machines to understand and respond to human language.

NLP is divided into two parts.

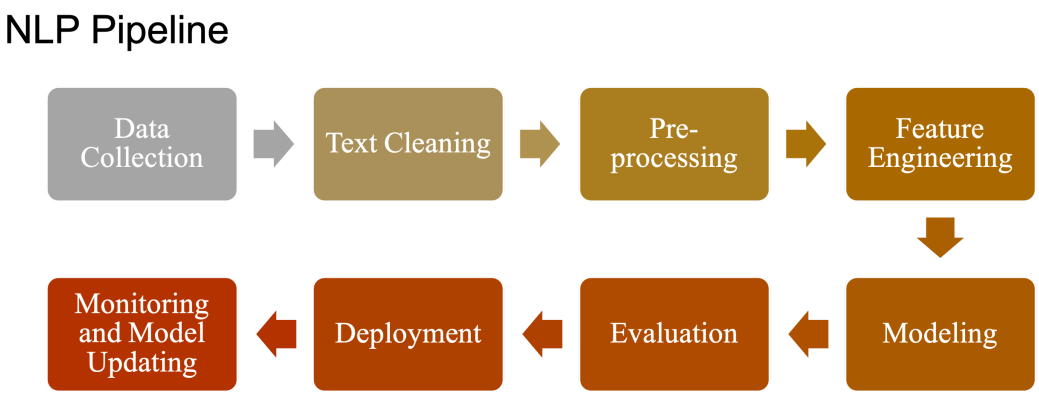
1. Natural Language Understanding - NLU
2. Natural Language Generation - NLG



Diff b/w NLU & NLG

Natural Language Understanding (NLU)	Natural Language Generation (NLG)
The transformation of Unstructured to Structured Data.	The transformation of Structured to Unstructured Data.
Generates data understandable by computers.	Generates data understandable by humans.

Natural Language Understanding



Data Collection

In [4]:
```
import nltk.corpus
```

We can collect data from excel files, csv files.

In [6]:
```
AI = '''Artificial Intelligence refers to the intelligence of machines. This is
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.'''
```

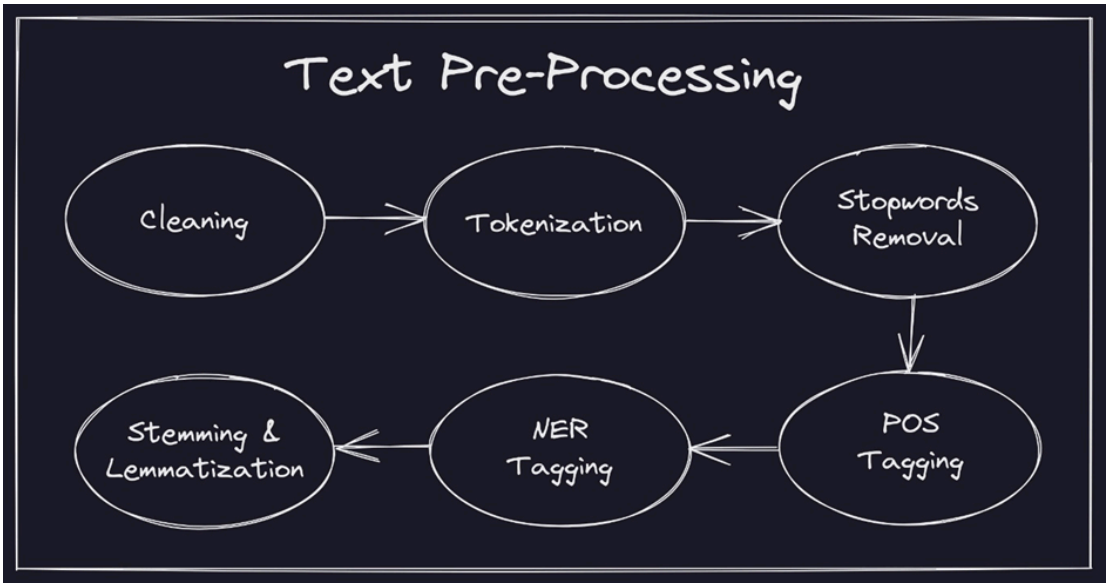
In [8]:
```
AI
```

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

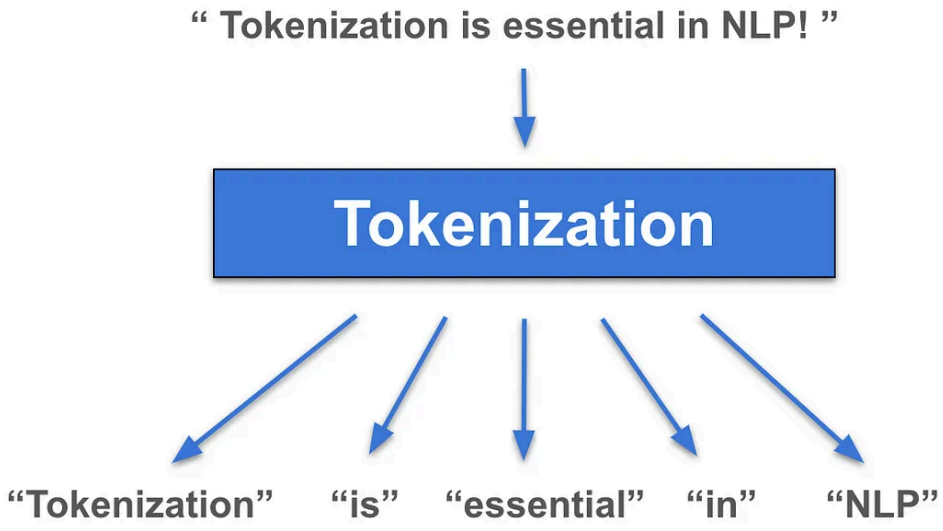
In [10]:
```
type(AI)
```

Out[10]:
```
str
```

Text Preprocessing



Tokenization



In [12]:
```
from nltk.tokenize import word_tokenize
```

In [14]:
```
AI_tokens = word_tokenize(AI)
AI_tokens
```

Out[14]:
```
['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 [16]:
```
len(AI_tokens)
```

Out[16]:
```
81
```

In [20]:
```
from nltk.tokenize import sent_tokenize
```

In [22]:
```
AI_sent = sent_tokenize(AI)
AI_sent
```

Out[22]:
```
['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 [24]:
```
len(AI_sent)
```

Out[24]:
```
6
```

In [26]:
```
from nltk.tokenize import blankline_tokenize # gives you how many paragraphs
AI_blank = blankline_tokenize(AI)
AI_blank
```

Out[26]:
```
['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 [28]:
```
len(AI_blank)
```

Out[28]: 1

In [30]:
```
from nltk.tokenize import WhitespaceTokenizer
wt = WhitespaceTokenizer().tokenize(AI)
wt
```

Out[30]:
```
['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 [32]:
```
len(wt)
```

Out[32]:
```
70
```

In [34]:
```
len(AI_tokens)
```

Out[34]:
```
81
```

In [36]:
```
s = 'Good apple cost $3.88 in hyderbad. Please buy me two of them. Thanks.'
s
```

Out[36]:
```
'Good apple cost $3.88 in hyderbad. Please buy me two of them. Thanks.'
```

In [38]:
```
from nltk.tokenize import wordpunct_tokenize
wd = wordpunct_tokenize(s)
wd
```

Out[38]:
```
['Good',
 'apple',
 'cost',
 '$',
 '3',
 '.',
 '88',
 'in',
 'hyderbad',
 '.',
 'Please',
 'buy',
 'me',
 'two',
 'of',
 'them',
 '.',
 'Thanks',
 '.']
```

In [40]:
```
w_p = wordpunct_tokenize(AI)
w_p
```

Out[40]:
```
['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 [42]:
```
len(w_p)
```

Out[42]: 85

In [70]:
```
from nltk.util import bigrams, trigrams, ngrams
```

In [44]:
```
string = 'hello the best and most beautifull thing in the world cannot be seen o
quotes_tokens = nltk.word_tokenize(string)
quotes_tokens
```

Out[44]:
```
['hello',
 'the',
 'best',
 'and',
 'most',
 'beautifull',
 'thing',
 'in',
 'the',
 'world',
 'can',
 'not',
 'be',
 'seen',
 'or',
 'even',
 'touched',
 ',',
 'they',
 'must',
 'be',
 'felt',
 'with',
 'heart']
```

In [46]:
```
string
```

Out[46]:
```
'hello the best and most beautifull thing in the world cannot be seen or even t
ouched,they must be felt with heart'
```

In [48]:
```
len(quotes_tokens)
```

Out[48]:
```
24
```

In [50]:
```
quotes_bigrams = list(nltk.bigrams(quotes_tokens))
quotes_bigrams
```

Out[50]:
```
[('hello', 'the'),
 ('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')]
```

In [52]:
```
quotes_trigrams = list(nltk.trigrams(quotes_tokens))
quotes_trigrams
```

Out[52]:
```
[('hello', 'the', 'best'),
 ('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')]
```

In [80]:
```
quotes_ngrams = list(nltk.ngrams(quotes_tokens,4))
quotes_ngrams
```

Out[80]:
```
[('hello', 'the', 'best', 'and'),
 ('the', 'best', 'and', 'most'),
 ('best', 'and', 'most', 'beautifull'),
 ('and', 'most', 'beautifull', 'thing'),
 ('most', 'beautifull', 'thing', 'in'),
 ('beautifull', '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')]
```

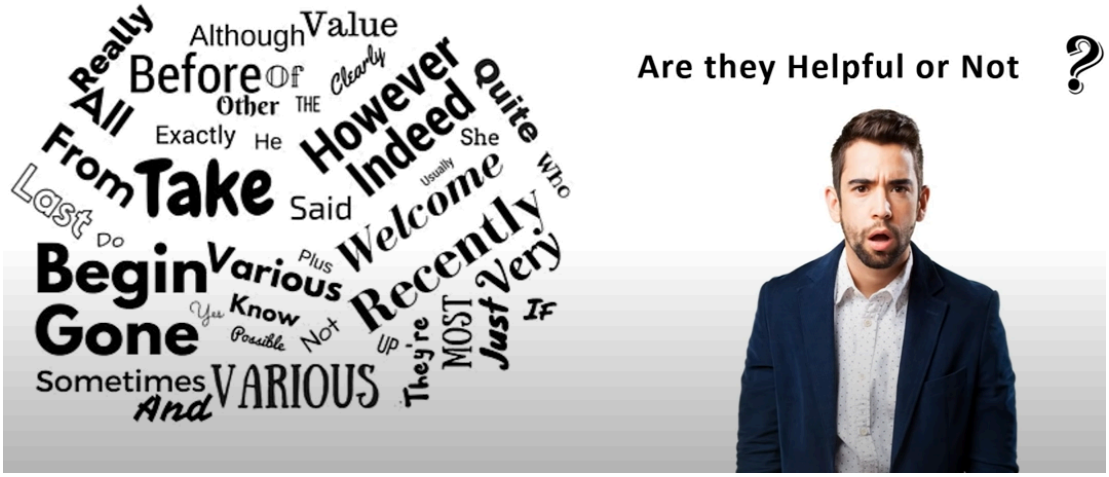
In [54]:
```
quotes_ngrams = list(nltk.ngrams(quotes_tokens,8))
quotes_ngrams
```

Out[54]:
```
[('hello', 'the', 'best', 'and', 'most', 'beautifull', 'thing', 'in'),
 ('the', 'best', 'and', 'most', 'beautifull', 'thing', 'in', 'the'),
 ('best', 'and', 'most', 'beautifull', 'thing', 'in', 'the', 'world'),
 ('and', 'most', 'beautifull', 'thing', 'in', 'the', 'world', 'can'),
 ('most', 'beautifull', 'thing', 'in', 'the', 'world', 'can', 'not'),
 ('beautifull', 'thing', 'in', 'the', 'world', 'can', 'not', 'be'),
 ('thing', 'in', 'the', 'world', 'can', 'not', 'be', 'seen'),
 ('in', 'the', 'world', 'can', 'not', 'be', 'seen', 'or'),
 ('the', 'world', 'can', 'not', 'be', 'seen', 'or', 'even'),
 ('world', 'can', 'not', 'be', 'seen', 'or', 'even', 'touched'),
 ('can', 'not', 'be', 'seen', 'or', 'even', 'touched', ','),
 ('not', 'be', 'seen', 'or', 'even', 'touched', ',', 'they'),
 ('be', 'seen', 'or', 'even', 'touched', ',', 'they', 'must'),
 ('seen', 'or', 'even', 'touched', ',', 'they', 'must', 'be'),
 ('or', 'even', 'touched', ',', 'they', 'must', 'be', 'felt'),
 ('even', 'touched', ',', 'they', 'must', 'be', 'felt', 'with'),
 ('touched', ',', 'they', 'must', 'be', 'felt', 'with', 'heart')]
```

Stop Words

Stop words are common words that appear in text data that have little to no meaning and are usually removed during text cleaning. For example, words like “the”, “a”, “an”, and “in” are commonly used stop words.

Stop Words



In [56]:
```
from nltk.corpus import stopwords
```

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

Out[58]:
```
['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 [60]:
```
len(stopwords.words('english'))
```

Out[60]: 179

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

Out[62]: 157

In [64]:
```
len(stopwords.words('german'))
```

Out[64]: 232

In [66]:
```
len(stopwords.words('chinese'))
```

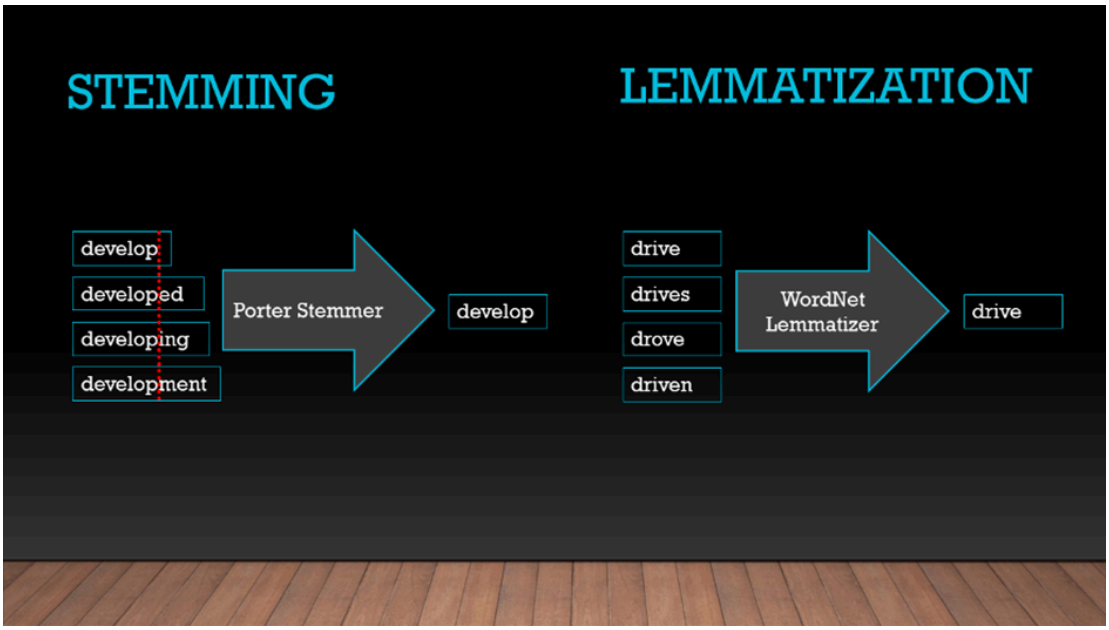
Out[66]: 841

In [68]:
```
# stopwords.words('hindi') --> research phase
```

Stemming and Lemmatization

What are Stemming and Lemmatization? Both techniques normalize text to prepare it for further processing:

Stemming reduces words to their root form by removing suffixes, even if the result isn't a valid word. Lemmatization brings words to their base or dictionary form, ensuring grammatical correctness.



Stemming

In [70]:
```
from nltk.stem import PorterStemmer
pst = PorterStemmer()
```

In [72]:
```
pst.stem('affection')
```

Out[72]: 'affect'

In [74]:
```
pst.stem('playing')
```

Out[74]: 'play'

In [76]:
```
pst.stem('maximum')
```

Out[76]: 'maximum'

In [78]:
```
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 [80]:
```
words_to_stem=['give','giving','given','gaved','thinking', 'loving','maximum']
# i am giving these different words to stem, using porter stemmer we get the out

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

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

In [82]:
```
from nltk.stem import LancasterStemmer
lst = LancasterStemmer()

for words in words_to_stem:
    print(words+ ' : ' + lst.stem(words))
```

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

In [84]:
```
from nltk.stem import SnowballStemmer
sbst = SnowballStemmer('english')

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

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

Lemmatization

In [86]:
```
from nltk.stem import wordnet
from nltk.stem import WordNetLemmatizer
word_lem = WordNetLemmatizer()
```

In [88]:
```
words_to_stem
```

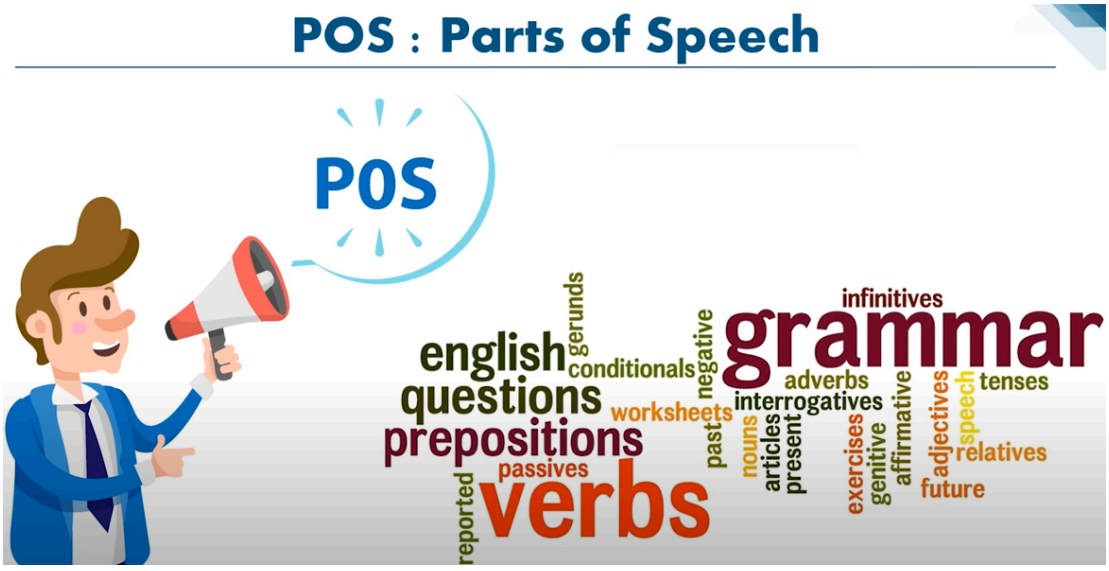
Out[88]:
```
['give', 'giving', 'given', 'gaved', 'thinking', 'loving', 'maximum']
```

In [90]:
```
for words in words_to_stem:
    print(words + ' : ' + word_lem.lemmatize(words))
```

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

POS --> Parts Of Speech

Part-of-Speech (POS) tagging is the process of labelling each word in a text corpus with its corresponding part of speech, such as noun, verb, adjective, adverb, etc. This can be useful for identifying the structure and meaning of a sentence, as well as for tasks like sentiment analysis and text classification.

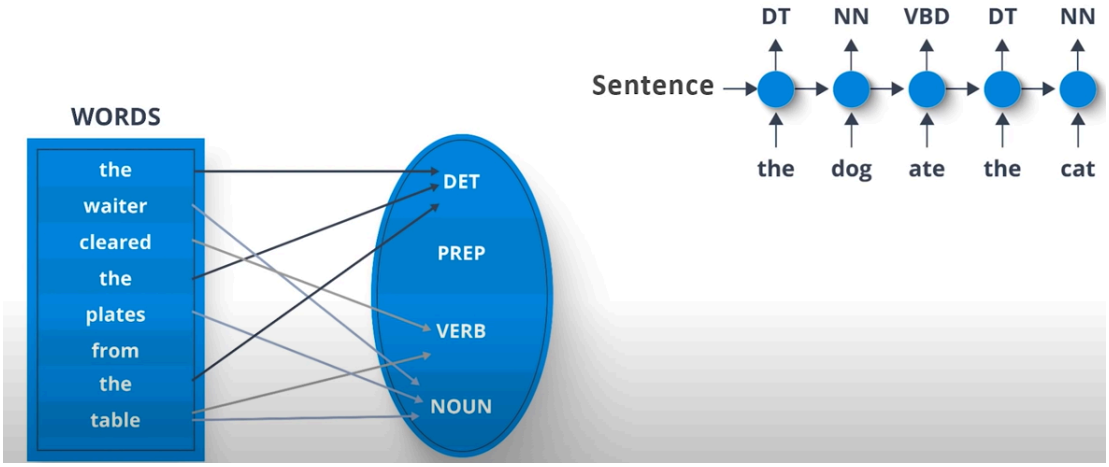


POS : Tags and Descriptions

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun

Tag	Description
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBR	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Whdeterminer
WP	Whpronoun
WP$	Possessive whpronoun
WRB	Whadverb

POS : Examples



In [92]:
```
sent = 'sam is a natural when it comes to drawing'
sent_tokens = word_tokenize(sent)
sent_tokens
```

Out[92]:
```
['sam', 'is', 'a', 'natural', 'when', 'it', 'comes', 'to', 'drawing']
```

In [94]:
```
for token in sent_tokens:
    print(nltk.pos_tag([token]))
```

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

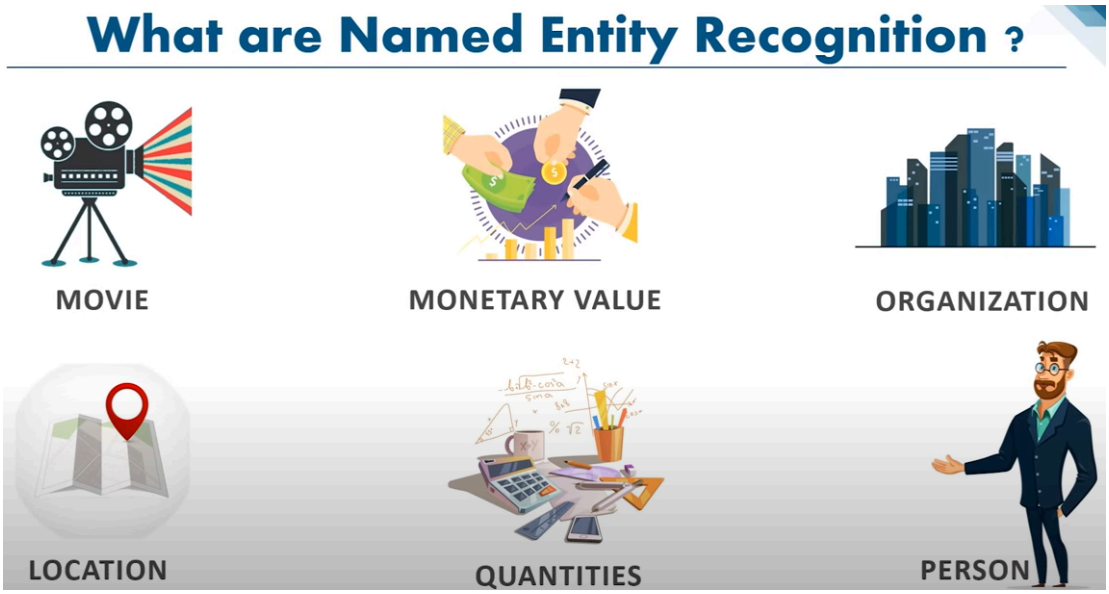
In [96]:
```
sent2 = 'john is eating a delicious cake'
sent2_tokens = word_tokenize(sent2)
```

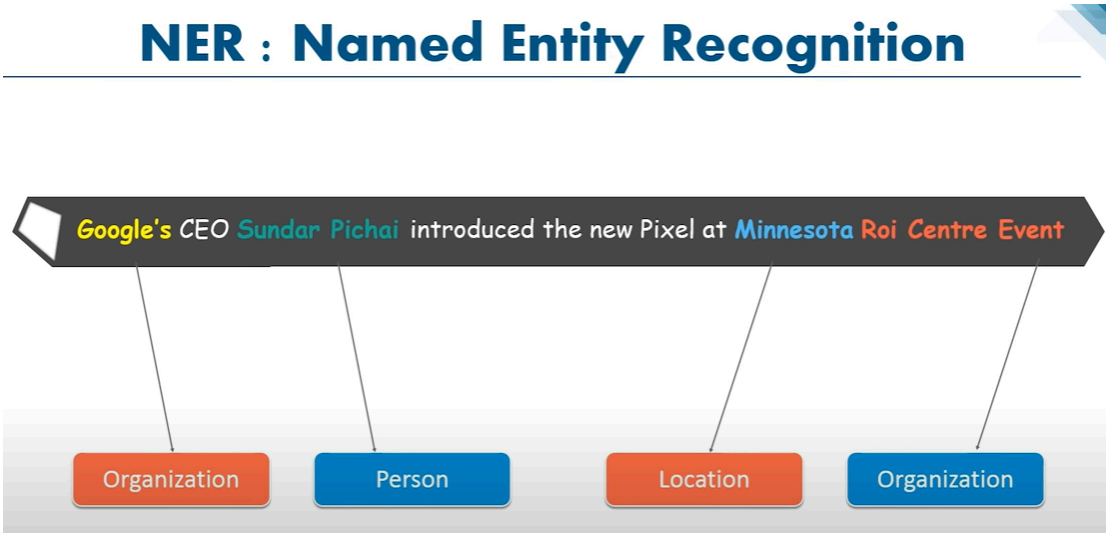
```
for token in sent2_tokens:
    print(nltk.pos_tag([token]))
```

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

NER --> Named Entity Recognition

Named Entity Recognition (NER) is the process of identifying and classifying the entities in text, such as people, organization, locations, and dates. This can be useful for tasks like information extraction and question answering.





In [98]:
```
from nltk import ne_chunk
```

In [100…
```
NE_sent = 'The US president stays in the WHITEHOUSE'
```

In [102…
```
NE_tokens = word_tokenize(NE_sent)
#after tokenize need to add the pos tags
```

```
NE_tokens
```

Out[102…
```
['The', 'US', 'president', 'stays', 'in', 'the', 'WHITEHOUSE']
```

In [104…
```
NE_tags = nltk.pos_tag(NE_tokens)
NE_tags
```

Out[104…
```
[('The', 'DT'),
 ('US', 'NNP'),
 ('president', 'NN'),
 ('stays', 'NNS'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('WHITEHOUSE', 'NNP')]
```

In [106…
```
#we are passin the NE_NER into ne_chunks function and lets see the outputs
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))
```

NLG - Natural Language Generation

NLG is a software process that turns structured data – converted by NLU and a (generally) non-linguistic representation of information – into a natural language output that humans can understand, usually in text format.

In [108…
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

In [110…
```
# !pip install WordCloud
```

In [112…
```
text = ('Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin
```

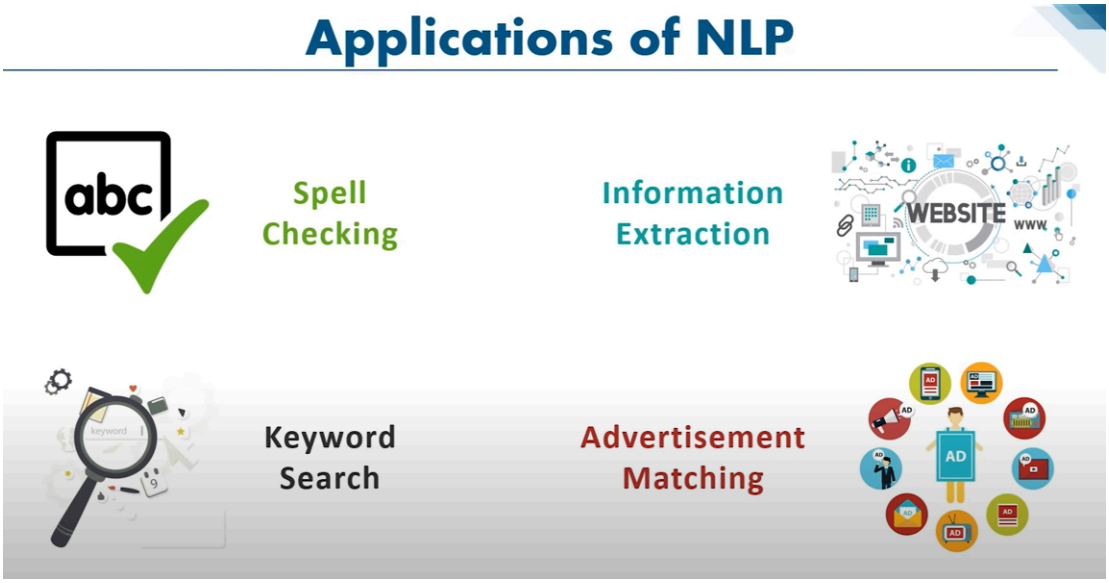
In [114…
```
text
```

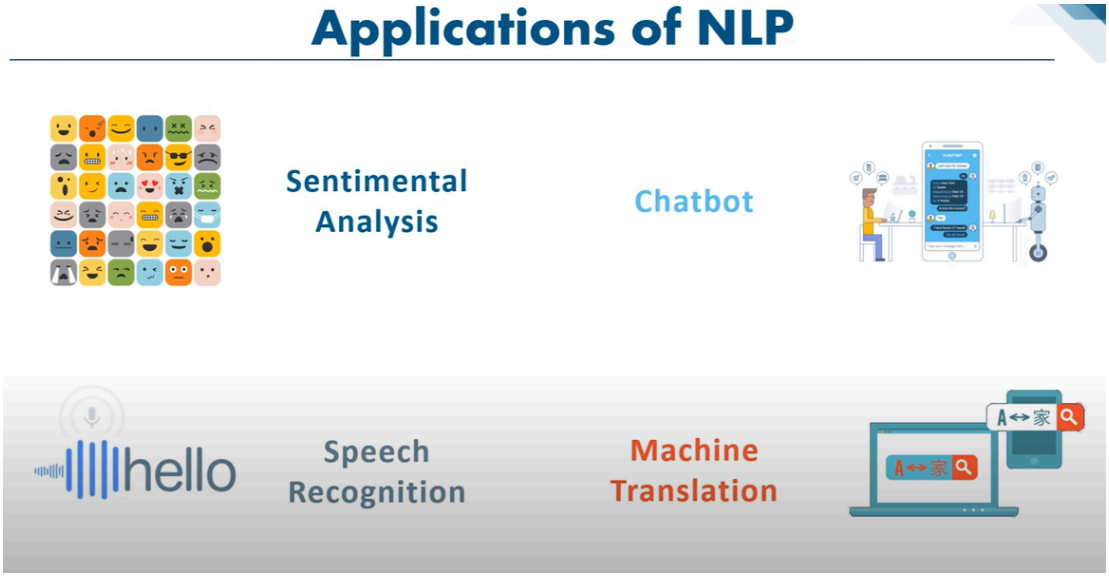
Out[114…
```
'Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Chart P
andas Datascience Wordcloud Spider Radar Parrallel Alpha Color Brewer Density S
catter Barplot Barplot Boxplot Violinplot Treemap Stacked Area Chart Chart Visu
alization Dataviz Donut Pie Time-Series Wordcloud Wordcloud Sankey Bubble'
```

In [116…
```
wordcloud = WordCloud(width = 420, height = 200, margin = 2, background_color =
```

In [118…
```
plt.imshow(wordcloud, interpolation = 'quadric')
plt.axis('off')
plt.margins(x = 0, y = 0)
plt.show()
```

Applications of NLP





Thank Tou