Step1: Sentence Segmentation Step2: Word Tokenization Step3: Stemming Step 4: Lemmatization Step 5: Identifying Stop Words Step 6: Dependency Parsing Step 7: POS tags Step 8: Named Entity Recognition (NER) Step 9: Chunking

In [1]:

```
import nltk.corpus
```

In [2]:

```
text = "In Brazil they drive on the right-hand side of the road. Brazil has a large coastli
from nltk.tokenize import word_tokenize# Passing the string text into word tokenize
token = word_tokenize(text)
token
```

Out[2]:

```
['In',
 'Brazil',
 'they',
 'drive',
 'on',
 'the',
 'right-hand',
 'side',
 'of',
 'the',
 'road',
 '.',
 'Brazil',
 'has',
 'a',
 'large',
 'coastli']
```

In [3]:

```
# finding the frequency distinct in the tokens
# Importing FreqDist library from nltk and passing token into FreqDist
from nltk.probability import FreqDist
fdist = FreqDist(token)
fdist
```

Out[3]:

```
FreqDist({'Brazil': 2, 'the': 2, 'In': 1, 'they': 1, 'drive': 1, 'on': 1, 'r
ight-hand': 1, 'side': 1, 'of': 1, 'road': 1, ...})
```

In [4]:

```
# To find the frequency of top 10 words
fdist1 = fdist.most_common(10)
fdist1
```

Out[4]:

```
[('Brazil', 2),
 ('the', 2),
 ('In', 1),
 ('they', 1),
 ('drive', 1),
 ('on', 1),
 ('right-hand', 1),
 ('side', 1),
 ('of', 1),
 ('road', 1)]
```

Stemming

In [5]:

```
# Importing Porterstemmer from nltk library
# Checking for the word ‘waiting’
from nltk.stem import PorterStemmer
pst = PorterStemmer()
pst.stem("waiting")
```

Out[5]:

```
'wait'
```

In [6]:

```
# Importing LancasterStemmer from nltk
from nltk.stem import LancasterStemmer
lst = LancasterStemmer()
stm = ["giving", "given", "given", "gave"]
for word in stm :
    print(word+ ":" +lst.stem(word))
```

```
giving:giv
given:giv
given:giv
gave:gav
```

Lemmatization

For example, lemmatization would correctly identify the base form of ‘caring’ to ‘care’, whereas, stemming would cutoff the ‘ing’ part and convert it to a car. Lemmatization can be implemented in python by using Wordnet Lemmatizer, Spacy Lemmatizer, TextBlob, Stanford CoreNLP

In [7]:

```
# Importing Lemmatizer library from nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
```

```
rocks : rock
corpora : corpus
```

Stop Words

“Stop words” are the most common words in a language like “the”, “a”, “at”, “for”, “above”, “on”, “is”, “all”. These words do not provide any meaning and are usually removed from texts. We can remove these stop words using nltk library

In [8]:

```
# importing stopwords from nltk library
from nltk import word_tokenize
from nltk.corpus import stopwords
a = set(stopwords.words('english'))
text = "Cristiano Ronaldo was born on February 5, 1985, in Funchal, Madeira, Portugal."
text1 = word_tokenize(text.lower())
print(text1)
stopwords = [x for x in text1 if x not in a]
print(stopwords)
```

```
['cristiano', 'ronaldo', 'was', 'born', 'on', 'february', '5', ',', '1985',
',', 'in', 'funchal', ',', 'madeira', ',', 'portugal', '.']
['cristiano', 'ronaldo', 'born', 'february', '5', ',', '1985', ',', 'funcha
l', ',', 'madeira', ',', 'portugal', '.']
```

Part of speech tagging (POS)

Part-of-speech tagging is used to assign parts of speech to each word of a given text (such as nouns, verbs, pronouns, adverbs, conjunction, adjectives, interjection) based on its definition and its context. There are many tools available for POS taggers and some of the widely used taggers are NLTK, Spacy, TextBlob, Standford CoreNLP, etc.

In [9]:

```
text = "vote to choose a particular man or a group (party) to represent them in parliament"
#Tokenize the text
tex = word_tokenize(text)
for token in tex:
    print(nltk.pos_tag([token]))
```

```
[('vote', 'NN')]
[('to', 'TO')]
[('choose', 'NN')]
[('a', 'DT')]
[('particular', 'JJ')]
[('man', 'NN')]
[('or', 'CC')]
[('a', 'DT')]
[('group', 'NN')]
[('(', '(')]
[('party', 'NN')]
[(')', ')')]
[('to', 'TO')]
[('represent', 'NN')]
[('them', 'PRP')]
[('in', 'IN')]
[('parliament', 'NN')]
```

Name entitity recognition

It is the process of detecting the named entities such as the person name, the location name, the company name, the quantities and the monetary value

In [10]:

```
!pip install ghostscript
```

```
Requirement already satisfied: ghostscript in c:\users\prathyu lachireddy\an
aconda3\lib\site-packages (0.7)
Requirement already satisfied: setuptools>=38.6.0 in c:\users\prathyu lachir
eddy\anaconda3\lib\site-packages (from ghostscript) (50.3.1.post20201107)
```

In [12]:

```
text = "Google's CEO Sundar Pichai introduced the new Pixel at Minnesota Roi Centre Event"
#importing chunk library from nltk
from nltk import ne_chunk# tokenize and POS Tagging before doing chunk
token = word_tokenize(text)
tags = nltk.pos_tag(token)
chunk = ne_chunk(tags)
chunk
```

```
The Ghostscript executable isn't found.
See http://web.mit.edu/ghostscript/www/Install.htm (http://web.mit.edu/ghost
script/www/Install.htm)
If you're using a Mac, you can try installing
https://docs.brew.sh/Installation (https://docs.brew.sh/Installation) then `
brew install ghostscript`

---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
~\anaconda3\lib\site-packages\nltk\tree.py in _repr_png_(self)
    797                     [
--> 798                         find_binary(
    799                             "gs",

~\anaconda3\lib\site-packages\nltk\internals.py in find_binary(name, path_to
_bin, env_vars, searchpath, binary_names, url, verbose)
    687 ):
--> 688     return next(
    689         find_binary_iter(

~\anaconda3\lib\site-packages\nltk\internals.py in find_binary_iter(name, pa
th_to_bin, env_vars, searchpath, binary_names, url, verbose)
    672     """
--> 673     for file in find_file_iter(
    674         path_to_bin or name, env_vars, searchpath, binary_names, url
, verbose

~\anaconda3\lib\site-packages\nltk\internals.py in find_file_iter(filename,
 env_vars, searchpath, file_names, url, verbose, finding_dir)
    631         div = "=" * 75
--> 632         raise LookupError("\n\n%s\n%s\n%s" % (div, msg, div))
    633

LookupError:

===========================================================================
NLTK was unable to find the gs file!
Use software specific configuration paramaters or set the PATH environment v
ariable.
===========================================================================

During handling of the above exception, another exception occurred:

LookupError                               Traceback (most recent call last)
~\anaconda3\lib\site-packages\IPython\core\formatters.py in __call__(self, o
bj)
    343             method = get_real_method(obj, self.print_method)
    344             if method is not None:
--> 345                 return method()
    346             return None
    347         else:
```

```
~\anaconda3\lib\site-packages\nltk\tree.py in _repr_png_(self)
    815                 )
    816                 print(pre_error_message, file=sys.stderr)
--> 817                 raise LookupError
    818
    819             with open(out_path, "rb") as sr:

LookupError:
```

Out[12]:

```
Tree('S', [Tree('PERSON', [('Google', 'NNP')]), ('’', 'NNP'), ('s', 'VBD'),
Tree('ORGANIZATION', [('CEO', 'NNP'), ('Sundar', 'NNP'), ('Pichai', 'NN
P')]), ('introduced', 'VBD'), ('the', 'DT'), ('new', 'JJ'), ('Pixel', 'NN
P'), ('at', 'IN'), Tree('ORGANIZATION', [('Minnesota', 'NNP'), ('Roi', 'NN
P'), ('Centre', 'NNP')]), ('Event', 'NNP')])
```

Chunking

Chunking means picking up individual pieces of information and grouping them into bigger pieces. In the context of NLP and text mining, chunking means a grouping of words or tokens into chunks.

In [12]:

```
text = "We saw the yellow dog"
token = word_tokenize(text)
tags = nltk.pos_tag(token)
reg = "NP: {<DT>?<JJ>*<NN>}"
a = nltk.RegexpParser(reg)
result = a.parse(tags)
print(result)
```

```
(S We/PRP saw/VBD (NP the/DT yellow/JJ dog/NN))
```

SENTIMENTAL ANALYSIS

In [13]:

```
from textblob import TextBlob
Feedback1 = "The food at the restaurant was awesome"
Feedback2 = " The food at ABC was very good"
Feedback3 = "The food was very bad"
blob1 = TextBlob(Feedback1)
blob2 = TextBlob(Feedback2)
blob3 = TextBlob(Feedback3)
print(blob1.sentiment)
print(blob2.sentiment)
print(blob3.sentiment)
```

```
Sentiment(polarity=1.0, subjectivity=1.0)
Sentiment(polarity=0.9099999999999999, subjectivity=0.7800000000000001)
Sentiment(polarity=-0.9099999999999998, subjectivity=0.8666666666666667)
```

Ayntonyms from WordNet

If you remember, we installed NLTK packages using nltk.download(). One of the packages was WordNet. WordNet is a database that is built for natural language processing. It includes groups of synonyms and a brief definition. You can get these definitions and examples for a given word like this:

In [13]:

```
from nltk.corpus import wordnet
syn = wordnet.synsets("pain")
print(syn[0].definition())
print(syn[0].examples())
```

```
a symptom of some physical hurt or disorder
['the patient developed severe pain and distension']
```

In [15]:

```
from nltk.corpus import wordnet
syn = wordnet.synsets("NLP")
print(syn[0].definition())
syn = wordnet.synsets("Python")
print(syn[0].definition())
```

```
the branch of information science that deals with natural language informati
on
large Old World boas
```

Get antonyms from WordNet .

You can get the antonyms words the same way, all you have to do is to check the lemmas before adding them to the array if it's an antonym or not.

In [16]:

```
from nltk.corpus import wordnet
antonyms = []
for syn in wordnet.synsets("big"):

    for l in syn.lemmas():
        if l.antonyms():
            antonyms.append(l.antonyms()[0].name())

print(antonyms)
```

```
['small', 'little', 'small']
```

In [1]:

```
from nltk import ne_chunk#tokenize and POS Tagging before doing chunk

text = "Google's CEO sundar pichai introducted"
```

VADER sentiment analysis

VADER Sentiment Analysis. VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule based sentiment analysis tool that is specifically attuned to sentiments expressed in social media, and works well on texts from other domains.

In [2]:

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

Date: - 28/05/21

In [3]:

```
import nltk
```

In [4]:

```
from nltk.tokenize import sent_tokenize, word_tokenize
```

In [5]:

```
text = '''Orange is a visual programming software package used for this domain. It has used
```

In [6]:

```
print(sent_tokenize(text))
```

```
['Orange is a visual programming software package used for this domain.', 'I
t has used widely, ranging from machine learning, data mining, and data anal
ysis, etc.', 'Orange tools (called widgets) are within the realm of simple d
ata visualization & pre-processing empirical evaluation of learning algorith
ms and predictive modelling.', 'Visual programming is implemented via a comb
ination in which workflows are designed by linking user-designed widgets.']
```

In [7]:

```
print(word_tokenize(text))
```

```
['Orange', 'is', 'a', 'visual', 'programming', 'software', 'package', 'use
d', 'for', 'this', 'domain', '.', 'It', 'has', 'used', 'widely', ',', 'rangi
ng', 'from', 'machine', 'learning', ',', 'data', 'mining', ',', 'and', 'dat
a', 'analysis', ',', 'etc', '.', 'Orange', 'tools', '(', 'called', 'widget
s', ')', 'are', 'within', 'the', 'realm', 'of', 'simple', 'data', 'visualiza
tion', '&', 'pre-processing', 'empirical', 'evaluation', 'of', 'learning',
'algorithms', 'and', 'predictive', 'modelling', '.', 'Visual', 'programmin
g', 'is', 'implemented', 'via', 'a', 'combination', 'in', 'which', 'workflow
s', 'are', 'designed', 'by', 'linking', 'user-designed', 'widgets', '.']
```

In [13]:

```
token = word_tokenize(text)
from nltk.probability import FreqDist
fdist = FreqDist(token)
fdist
```

Out[13]:

```
FreqDist({'.': 4, ',': 4, 'data': 3, 'Orange': 2, 'is': 2, 'a': 2, 'programm
ing': 2, 'used': 2, 'learning': 2, 'and': 2, ...})
```

In [16]:

```
from nltk import word_tokenize
from nltk.corpus import stopwords
a = set(stopwords.words('english'))
text1 = word_tokenize(text.lower())
print(text1)
stopwords = [x for c in text if x not in a]
print(stopwords)
```

```
['orange', 'is', 'a', 'visual', 'programming', 'software', 'package', 'use
d', 'for', 'this', 'domain', '.', 'it', 'has', 'used', 'widely', ',', 'rangi
ng', 'from', 'machine', 'learning', ',', 'data', 'mining', ',', 'and', 'dat
a', 'analysis', ',', 'etc', '.', 'orange', 'tools', '(', 'called', 'widget
s', ')', 'are', 'within', 'the', 'realm', 'of', 'simple', 'data', 'visualiza
tion', '&', 'pre-processing', 'empirical', 'evaluation', 'of', 'learning',
'algorithms', 'and', 'predictive', 'modelling', '.', 'visual', 'programmin
g', 'is', 'implemented', 'via', 'a', 'combination', 'in', 'which', 'workflow
s', 'are', 'designed', 'by', 'linking', 'user-designed', 'widgets', '.']

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-16-0462295fd555> in <module>
      4 text1 = word_tokenize(text.lower())
      5 print(text1)
----> 6 stopwords = [x for c in text if x not in a]
      7 print(stopwords)

<ipython-input-16-0462295fd555> in <listcomp>(.0)
      4 text1 = word_tokenize(text.lower())
      5 print(text1)
----> 6 stopwords = [x for c in text if x not in a]
      7 print(stopwords)

NameError: name 'x' is not defined
```

In [17]:

```
#stemming

from nltk.stem import PorterStemmer
stemming = PorterStemmer()
stemming.stem(text)
```

Out[17]:

```
'orange is a visual programming software package used for this domain. it ha
s used widely, ranging from machine learning, data mining, and data analysi
s, etc. orange tools (called widgets) are within the realm of simple data vi
sualization & pre-processing empirical evaluation of learning algorithms and
predictive modelling. visual programming is implemented via a combination in
which workflows are designed by linking user-designed widgets.'
```

In [18]:

```
#Lemmantization

from nltk.stem import WordNetLemmatizer
lemmatization = WordNetLemmatizer()
print(text)
```

```
Orange is a visual programming software package used for this domain. It has
used widely, ranging from machine learning, data mining, and data analysis,
etc. Orange tools (called widgets) are within the realm of simple data visua
lization & pre-processing empirical evaluation of learning algorithms and pr
edictive modelling. Visual programming is implemented via a combination in w
hich workflows are designed by linking user-designed widgets.
```

In [19]:

```
text2 = word_tokenize(text)
for token in text2:
    print(nltk.pos_tag([token]))
```

```
[('Orange', 'NN')]
[('is', 'VBZ')]
[('a', 'DT')]
[('visual', 'JJ')]
[('programming', 'VBG')]
[('software', 'NN')]
[('package', 'NN')]
[('used', 'VBN')]
[('for', 'IN')]
[('this', 'DT')]
[('domain', 'NN')]
[('.', '.')]
[('It', 'PRP')]
[('has', 'VBZ')]
[('used', 'VBN')]
[('widely', 'RB')]
[(',', ',')]
[('ranging', 'VBG')]
[('from', 'IN')]
[('machine', 'NN')]
[('learning', 'VBG')]
[(',', ',')]
[('data', 'NNS')]
[('mining', 'NN')]
[(',', ',')]
[('and', 'CC')]
[('data', 'NNS')]
[('analysis', 'NN')]
[(',', ',')]
[('etc', 'NN')]
[('.', '.')]
[('Orange', 'NN')]
[('tools', 'NNS')]
[('(', '(')]
[('called', 'VBN')]
[('widgets', 'NNS')]
[(')', ')')]
[('are', 'VBP')]
[('within', 'IN')]
[('the', 'DT')]
[('realm', 'NN')]
[('of', 'IN')]
[('simple', 'NN')]
[('data', 'NNS')]
[('visualization', 'NN')]
[('&', 'CC')]
[('pre-processing', 'NN')]
[('empirical', 'JJ')]
[('evaluation', 'NN')]
[('of', 'IN')]
[('learning', 'VBG')]
[('algorithms', 'NN')]
[('and', 'CC')]
[('predictive', 'NN')]
[('modelling', 'VBG')]
```

```
[('.', '.')]
[('Visual', 'JJ')]
[('programming', 'VBG')]
[('is', 'VBZ')]
[('implemented', 'VBN')]
[('via', 'IN')]
[('a', 'DT')]
[('combination', 'NN')]
[('in', 'IN')]
[('which', 'WDT')]
[('workflows', 'NNS')]
[('are', 'VBP')]
[('designed', 'VBN')]
[('by', 'IN')]
[('linking', 'VBG')]
[('user-designed', 'JJ')]
[('widgets', 'NNS')]
[('.', '.')]
```

In [27]:

```
from nltk import ne_chunk
text2 = "sam is driving a car"
token = word_tokenize(text2)
tags = nltk.pos_tag(token)
chunk=ne_chunk(tags)
```

In [28]:

```
chunk
```

```
The Ghostscript executable isn't found.
See http://web.mit.edu/ghostscript/www/Install.htm (http://web.mit.edu/ghost
script/www/Install.htm)
If you're using a Mac, you can try installing
https://docs.brew.sh/Installation (https://docs.brew.sh/Installation) then `
brew install ghostscript`

---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
~\anaconda3\lib\site-packages\nltk\tree.py in _repr_png_(self)
    797                     [
--> 798                         find_binary(
    799                             "gs",

~\anaconda3\lib\site-packages\nltk\internals.py in find_binary(name, path_to
_bin, env_vars, searchpath, binary_names, url, verbose)
    687 ):
--> 688     return next(
    689         find_binary_iter(

~\anaconda3\lib\site-packages\nltk\internals.py in find_binary_iter(name, pa
th_to_bin, env_vars, searchpath, binary_names, url, verbose)
    672     """
--> 673     for file in find_file_iter(
    674         path_to_bin or name, env_vars, searchpath, binary_names, url
, verbose

~\anaconda3\lib\site-packages\nltk\internals.py in find_file_iter(filename,
 env_vars, searchpath, file_names, url, verbose, finding_dir)
    631         div = "=" * 75
--> 632         raise LookupError("\n\n%s\n%s\n%s" % (div, msg, div))
    633

LookupError:

===========================================================================
NLTK was unable to find the gs file!
Use software specific configuration paramaters or set the PATH environment v
ariable.
===========================================================================

During handling of the above exception, another exception occurred:

LookupError                               Traceback (most recent call last)
~\anaconda3\lib\site-packages\IPython\core\formatters.py in __call__(self, o
bj)
    343             method = get_real_method(obj, self.print_method)
    344             if method is not None:
--> 345                 return method()
    346             return None
    347         else:

~\anaconda3\lib\site-packages\nltk\tree.py in _repr_png_(self)
    815                 )
    816                 print(pre_error_message, file=sys.stderr)
--> 817                 raise LookupError
    818
```

```
    819             with open(out_path, "rb") as sr:

LookupError:
```

Out[28]:

```
Tree('S', [('sam', 'NN'), ('is', 'VBZ'), ('driving', 'VBG'), ('a', 'DT'),
('car', 'NN')])
```

In [29]:

```
from textblob import TextBlob
blob = TextBlob(text)
print(blob.sentiment)
```

```
Sentiment(polarity=0.0, subjectivity=0.17142857142857143)
```

In [ ]:

```

```