Natural Language Processing (NLP)

In [1]:
```
#Open the text file :
text_file = open("Natural_Language_Processing_Text.txt")

#Read the data :
text = text_file.read()

#Datatype of the data read :
print (type(text))
```
```
<class 'str'>
```

In [2]:
```
#Print the text :
print(text)
print("\n")
#Length of the text :
print (len(text))
```
```
Once upon a time there was an old mother pig who had three little pigs and not enough food to feed them. So whe
n they were old enough, she sent them out into the world to seek their fortunes.

The first little pig was very lazy. He didn't want to work at all and he built his house out of straw. The seco
nd little pig worked a little bit harder but he was somewhat lazy too and he built his house out of sticks. The
n, they sang and danced and played together the rest of the day.

The third little pig worked hard all day and built his house with bricks. It was a sturdy house complete with a
fine fireplace and chimney. It looked like it could withstand the strongest winds.


675
```

In [3]:
```
#Import required libraries :
import nltk
from nltk import sent_tokenize
from nltk import word_tokenize


nltk.download("popular")
```
```
[nltk_data] Downloading collection 'popular'
[nltk_data]    |
[nltk_data]    | Downloading package cmudict to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package gazetteers to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package gazetteers is already up-to-date!
[nltk_data]    | Downloading package genesis to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package inaugural to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package inaugural is already up-to-date!
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package names to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package shakespeare to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package shakespeare is already up-to-date!
[nltk_data]    | Downloading package stopwords to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package treebank to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package twitter_samples is already up-to-date!
[nltk_data]    | Downloading package omw to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package omw is already up-to-date!
[nltk_data]    | Downloading package wordnet to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package punkt to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package snowball_data is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection popular
```
Out[3]: True

In [4]:
```
#Tokenize the text by sentences :
sentences = sent_tokenize(text)

#How many sentences are there? :
print (len(sentences))
```
```
9
```

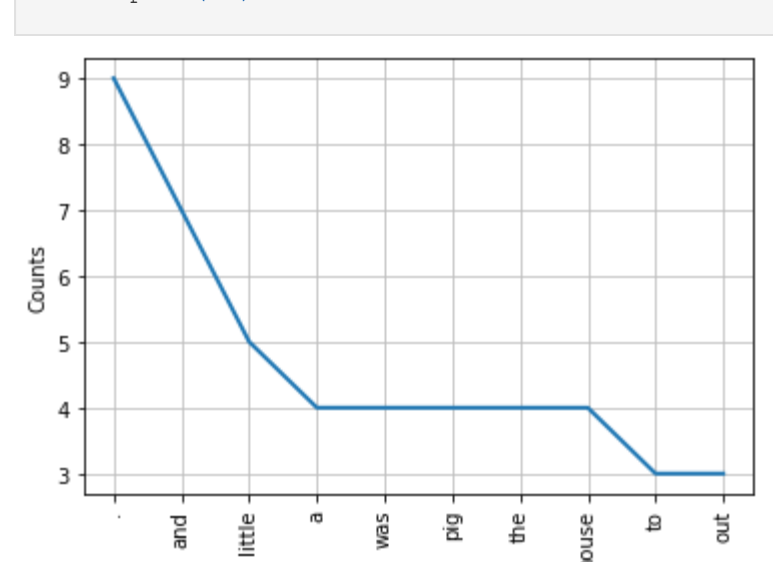
In [5]:
```
#Print the sentences :
#print(sentences)
sentences
```
Out[5]:
```
['Once upon a time there was an old mother pig who had three little pigs and not enough food to feed them.',
 'So when they were old enough, she sent them out into the world to seek their fortunes.',
 'The first little pig was very lazy.',
 "He didn't want to work at all and he built his house out of straw.",
 'The second little pig worked a little bit harder but he was somewhat lazy too and he built his house out of s
ticks.',
 'Then, they sang and danced and played together the rest of the day.',
 'The third little pig worked hard all day and built his house with bricks.',
 'It was a sturdy house complete with a fine fireplace and chimney.',
 'It looked like it could withstand the strongest winds.']
```

In [6]:
```
#Tokenize the text with words :
words = word_tokenize(text)

#How many words are there? :
print (len(words))
print("\n")
```
```
144
```

In [7]:
```
#Print words :
print (words)
```
```
['Once', 'upon', 'a', 'time', 'there', 'was', 'an', 'old', 'mother', 'pig', 'who', 'had', 'three', 'little', 'p
igs', 'and', 'not', 'enough', 'food', 'to', 'feed', 'them', '.', 'So', 'when', 'they', 'were', 'old', 'enough',
',', 'she', 'sent', 'them', 'out', 'into', 'the', 'world', 'to', 'seek', 'their', 'fortunes', '.', 'The', 'firs
t', 'little', 'pig', 'was', 'very', 'lazy', '.', 'He', 'did', "n't", 'want', 'to', 'work', 'at', 'all', 'and',
'he', 'built', 'his', 'house', 'out', 'of', 'straw', '.', 'The', 'second', 'little', 'pig', 'worked', 'a', 'lit
tle', 'bit', 'harder', 'but', 'he', 'was', 'somewhat', 'lazy', 'too', 'and', 'he', 'built', 'his', 'house', 'ou
t', 'of', 'sticks', '.', 'Then', ',', 'they', 'sang', 'and', 'danced', 'and', 'played', 'together', 'the', 'res
t', 'of', 'the', 'day', '.', 'The', 'third', 'little', 'pig', 'worked', 'hard', 'all', 'day', 'and', 'built',
'his', 'house', 'with', 'bricks', '.', 'It', 'was', 'a', 'sturdy', 'house', 'complete', 'with', 'a', 'fine', 'f
ireplace', 'and', 'chimney', '.', 'It', 'looked', 'like', 'it', 'could', 'withstand', 'the', 'strongest', 'wind
s', '.']
```

In [8]:
```
#Import required libraries :
from nltk.probability import FreqDist

#Find the frequency :
fdist = FreqDist(words)

fdist
```
Out[8]:
```
FreqDist({'.': 9, 'and': 7, 'little': 5, 'a': 4, 'was': 4, 'pig': 4, 'the': 4, 'house': 4, 'to': 3, 'out': 3,
...})
```

In [9]:
```
#Print 10 most common words :
fdist.most_common(10)
```
Out[9]:
```
[('.', 9),
 ('and', 7),
 ('little', 5),
 ('a', 4),
 ('was', 4),
 ('pig', 4),
 ('the', 4),
 ('house', 4),
 ('to', 3),
 ('out', 3)]
```

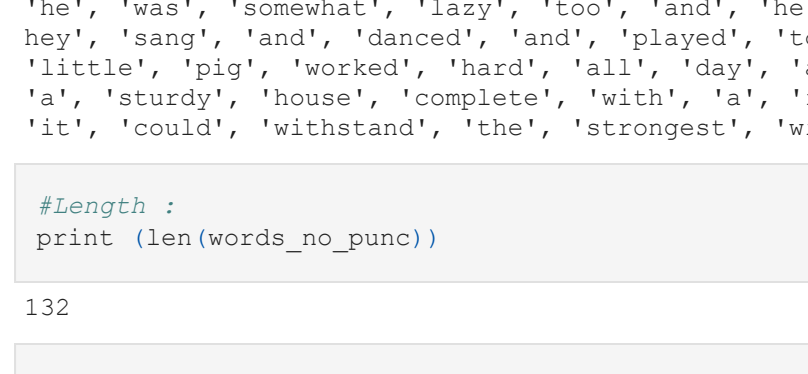
In [10]:
```
#Plot the graph for fdist :
import matplotlib.pyplot as plt

fdist.plot(10)
```


Out[10]: <AxesSubplot:xlabel='Samples', ylabel='Counts'>

In [11]:
```
#Empty list to store words:
words_no_punc = []

#Removing punctuation marks :
for w in words:
    if w.isalpha():
        words_no_punc.append(w.lower())

#Print the words without punctution marks :
print (words_no_punc)
```
```
['once', 'upon', 'a', 'time', 'there', 'was', 'an', 'old', 'mother', 'pig', 'who', 'had', 'three', 'little', 'p
igs', 'and', 'not', 'enough', 'food', 'to', 'feed', 'them', 'so', 'when', 'they', 'were', 'old', 'enough', 'sh
e', 'sent', 'them', 'out', 'into', 'the', 'world', 'to', 'seek', 'their', 'fortunes', 'the', 'first', 'little',
'pig', 'was', 'very', 'lazy', 'he', 'did', 'want', 'to', 'work', 'at', 'all', 'and', 'he', 'built', 'his', 'hou
se', 'out', 'of', 'straw', 'the', 'second', 'little', 'pig', 'worked', 'a', 'little', 'bit', 'harder', 'but',
'he', 'was', 'somewhat', 'lazy', 'too', 'and', 'he', 'built', 'his', 'house', 'out', 'of', 'sticks', 'then', 't
hey', 'sang', 'and', 'danced', 'and', 'played', 'together', 'the', 'rest', 'of', 'the', 'day', 'the', 'third',
'little', 'pig', 'worked', 'hard', 'all', 'day', 'and', 'built', 'his', 'house', 'with', 'bricks', 'it', 'was',
'a', 'sturdy', 'house', 'complete', 'with', 'a', 'fine', 'fireplace', 'and', 'chimney', 'it', 'looked', 'like',
'it', 'could', 'withstand', 'the', 'strongest', 'winds']
```

In [12]:
```
#Length :
print (len(words_no_punc))
```
```
132
```

In [13]:
```
#Frequency distribution :
fdist = FreqDist(words_no_punc)

fdist.most_common(10)
```
Out[13]:
```
[('and', 7),
 ('the', 7),
 ('little', 5),
 ('a', 4),
 ('was', 4),
 ('pig', 4),
 ('he', 4),
 ('house', 4),
 ('to', 3),
 ('out', 3)]
```

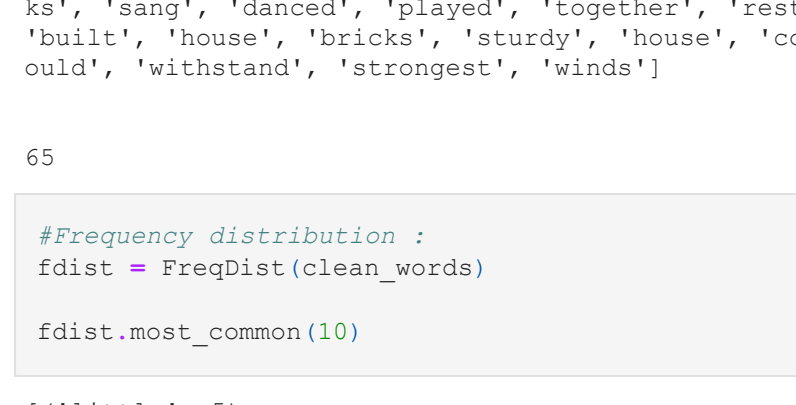
In [14]:
```
#Plot the most common words on grpah:

fdist.plot(10)
```


Out[14]: <AxesSubplot:xlabel='Samples', ylabel='Counts'>

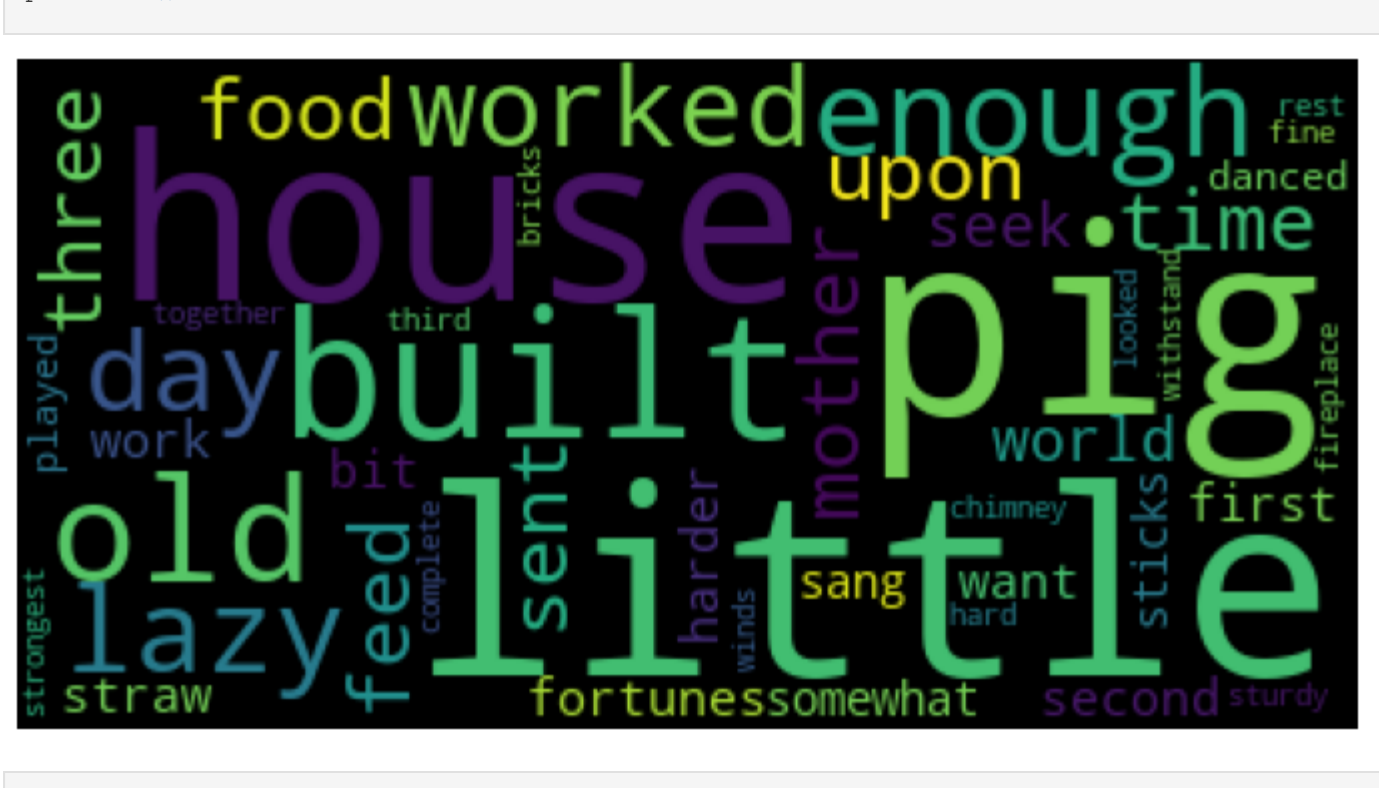
In [15]:
```
from nltk.corpus import stopwords

#List of stopwords
stopwords = stopwords.words("english")
print(stopwords)
```
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'yo
ur', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'hersel
f', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'wh
om', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'hav
e', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'becaus
e', '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', 'u
nder', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'bot
h', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 't
han', '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', "m
ustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'w
on', "won't", 'wouldn', "wouldn't"]
```

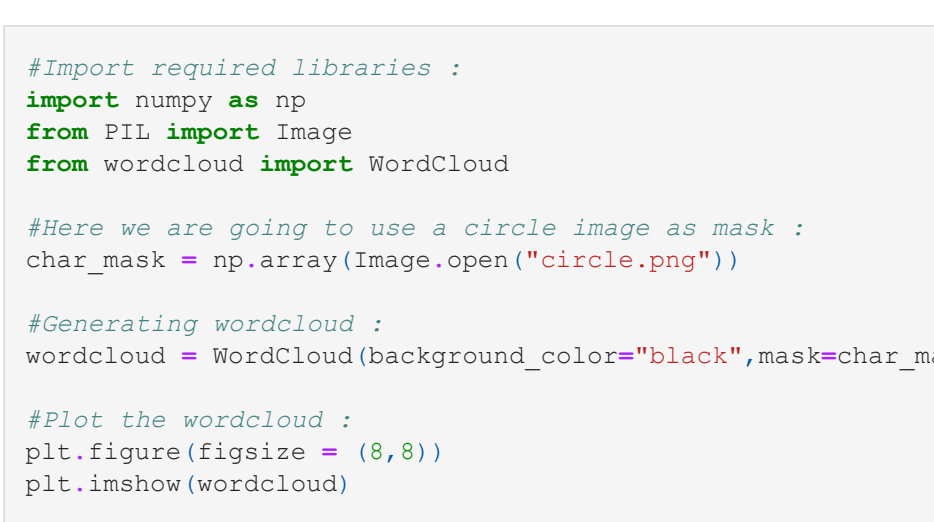
In [16]:
```
#Empty list to store clean words :
clean_words = []

for w in words_no_punc:
    if w not in stopwords:
        clean_words.append(w)

print(clean_words)
print("\n")
print(len(clean_words))
```
```
['upon', 'time', 'old', 'mother', 'pig', 'three', 'little', 'pigs', 'enough', 'food', 'feed', 'old', 'enough',
'sent', 'world', 'seek', 'fortunes', 'first', 'little', 'pig', 'lazy', 'want', 'work', 'built', 'house', 'stra
w', 'second', 'little', 'pig', 'worked', 'little', 'bit', 'harder', 'somewhat', 'lazy', 'built', 'house', 'stic
ks', 'sang', 'danced', 'played', 'together', 'rest', 'day', 'third', 'little', 'pig', 'worked', 'hard', 'day',
'built', 'house', 'bricks', 'sturdy', 'house', 'complete', 'fine', 'fireplace', 'chimney', 'looked', 'like', 'c
ould', 'withstand', 'strongest', 'winds']


65
```

In [17]:
```
#Frequency distribution :
fdist = FreqDist(clean_words)

fdist.most_common(10)
```
Out[17]:
```
[('little', 5),
 ('pig', 4),
 ('house', 4),
 ('built', 3),
 ('old', 2),
 ('enough', 2),
 ('lazy', 2),
 ('worked', 2),
 ('day', 2),
 ('upon', 1)]
```

In [18]:
```
#Plot the most common words on grpah:

fdist.plot(10)
```


Out[18]: <AxesSubplot:xlabel='Samples', ylabel='Counts'>

In [19]:
```
#Library to form wordcloud :
from wordcloud import WordCloud

#Library to plot the wordcloud :
import matplotlib.pyplot as plt

#Generating the wordcloud :
wordcloud = WordCloud().generate(text)

#Plot the wordcloud :
plt.figure(figsize = (12, 12))
plt.imshow(wordcloud)

#To remove the axis value :
plt.axis("off")
plt.show()
```


In [20]:
```
#Import required libraries :
import numpy as np
from PIL import Image
from wordcloud import WordCloud

#Here we are going to use a circle image as mask :
char_mask = np.array(Image.open("circle.png"))

#Generating wordcloud :
wordcloud = WordCloud(background_color="black",mask=char_mask).generate(text)

#Plot the wordcloud :
plt.figure(figsize = (8,8))
plt.imshow(wordcloud)

#To remove the axis value :
plt.axis("off")
plt.show()
```


In [21]:
```
#Stemming Example :

#Import stemming library :
from nltk.stem import PorterStemmer

porter = PorterStemmer()

#Word-list for stemming :
word_list = ["Study","Studying","Studies","Studied"]

for w in word_list:
    print(porter.stem(w))
```
```
studi
studi
studi
studi
```

In [22]:
```
#Stemming Example :

#Import stemming library :
from nltk.stem import PorterStemmer

porter = PorterStemmer()

#Word-list for stemming :
word_list = ["studies","leaves","decreases","plays"]

for w in word_list:
    print(porter.stem(w))
```
```
studi
leav
decreas
play
```

In [23]:
```
#Stemming Example :

#Import stemming library :
from nltk.stem import SnowballStemmer

snowball = SnowballStemmer("english")

#Word-list for stemming :
word_list = ["Study","Studying","Studies","Studied"]

for w in word_list:
    print(snowball.stem(w))
```
```
studi
studi
studi
studi
```

In [24]:
```
#Stemming Example :

#Import stemming library :
from nltk.stem import SnowballStemmer

#Print languages supported :
SnowballStemmer.languages
```
Out[24]:
```
('arabic',
 'danish',
 'dutch',
 'english',
 'finnish',
 'french',
 'german',
 'hungarian',
 'italian',
 'norwegian',
 'porter',
 'portuguese',
 'romanian',
 'russian',
 'spanish',
 'swedish')
```

In [25]:
```
from nltk import WordNetLemmatizer

lemma = WordNetLemmatizer()
word_list = ["Study","Studying","Studies","Studied"]

for w in word_list:
    print(lemma.lemmatize(w ,pos="v"))
```
```
Study
Studying
Studies
Studied
```

In [26]:
```
from nltk import WordNetLemmatizer

lemma = WordNetLemmatizer()
word_list = ["am","is","are","was","were"]

for w in word_list:
    print(lemma.lemmatize(w ,pos="v")) # POS-Tagging PARTS OF SPEECH, v- VERB
```
```
be
be
be
be
be
```

In [27]:
```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

print(stemmer.stem('studies'))
```
```
studi
```

In [28]:
```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize('studies'))
```
```
study
```

In [29]:
```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize('loves'))
```
```
love
```

In [30]:
```
from nltk import WordNetLemmatizer

lemma = WordNetLemmatizer()
word_list = ["studies","leaves","decreases","plays"]

for w in word_list:
    print(lemma.lemmatize(w))
```
```
study
leaf
decrease
play
```

In [31]:
```
#PoS tagging :
tag = nltk.pos_tag(["Studying","Study"])
print (tag)
```
```
[('Studying', 'VBG'), ('Study', 'NN')]
```

In [32]:
```
#PoS tagging example :

sentence = "A very beautiful young lady is walking on the beach"

#Tokenizing words :
tokenized_words = word_tokenize(sentence)

for words in tokenized_words:
    tagged_words = nltk.pos_tag(tokenized_words)

tagged_words
```
Out[32]:
```
[('A', 'DT'),
 ('very', 'RB'),
 ('beautiful', 'JJ'),
 ('young', 'JJ'),
 ('lady', 'NN'),
 ('is', 'VBZ'),
 ('walking', 'VBG'),
 ('on', 'IN'),
 ('the', 'DT'),
 ('beach', 'NN')]
```

In [33]:
```
#Extracting Noun Phrase from text :

# ? - optional character
# * - 0 or more repetations
grammar = "NP : {<DT>?<JJ>*<NN>} "
import matplotlib.pyplot as plt
#Creating a parser :
parser = nltk.RegexpParser(grammar)

#Parsing text :
output = parser.parse(tagged_words)
print (output)

#To visualize :
#output.draw()
```
```
(S
  A/DT
  very/RB
  (NP beautiful/JJ young/JJ lady/NN)
  is/VBZ
  walking/VBG
  on/IN
  (NP the/DT beach/NN))
```

In [34]:
```
#Chinking example :
# * - 0 or more repetations
# + - 1 or more repetations

#Here we are taking the whole string and then
#excluding adjectives from that chunk.

grammar = r""" NP: {<.*>+}
               }<JJ>+{"""

#Creating parser :
parser = nltk.RegexpParser(grammar)

#parsing string :
output = parser.parse(tagged_words)
print(output)

#To visualize :
#output.draw()
```
```
(S
  (NP A/DT very/RB)
  beautiful/JJ
  young/JJ
  (NP lady/NN is/VBZ walking/VBG on/IN the/DT beach/NN))
```

In [37]:
```
#Sentence for NER :
sentence = "Mr. Smith made a deal on a beach of Switzerland near WHO."

#Tokenizing words :
tokenized_words = word_tokenize(sentence)

#PoS tagging :
for w in tokenized_words:
    tagged_words = nltk.pos_tag(tokenized_words)

#print (tagged_words)

#Named Entity Recognition :
N_E_R = nltk.ne_chunk(tagged_words,binary=False)
print(N_E_R)
```
```
(S
  (PERSON Mr./NNP)
  (PERSON Smith/NNP)
  made/VBD
  a/DT
  deal/NN
  on/IN
  a/DT
  beach/NN
  of/IN
  (GPE Switzerland/NNP)
  near/IN
  (ORGANIZATION WHO/NNP)
  ./.)
```

In [38]:
```
#Sentence for NER :
sentence = "Mr. Smith made a deal on a beach of Switzerland near WHO."

#Tokenizing words :
tokenized_words = word_tokenize(sentence)

#PoS tagging :
for w in tokenized_words:
    tagged_words = nltk.pos_tag(tokenized_words)

#print (tagged_words)

#Named Entity Recognition :
N_E_R = nltk.ne_chunk(tagged_words,binary=True)

print(N_E_R)

#To visualize :
#N_E_R.draw()
```
```
(S
  (NE Mr./NNP Smith/NNP)
  made/VBD
  a/DT
  deal/NN
  on/IN
  a/DT
  beach/NN
  of/IN
  (NE Switzerland/NNP)
  near/IN
  (NE WHO/NNP)
  ./.)
```

In [39]:
```
#Import wordnet : cognitive synonyms called 'synsets',
from nltk.corpus import wordnet

for words in wordnet.synsets("Fun"):
    print(words)
```
```
Synset('fun.n.01')
Synset('fun.n.02')
Synset('fun.n.03')
Synset('playfulness.n.02')
```

In [40]:
```
#Word meaning with definitions :
for words in wordnet.synsets("Fun"):
    print(words.name())
    print(words.definition())
    print(words.examples())

    for lemma in words.lemmas():
        print(lemma)
    print("\n")
```
```
fun.n.01
activities that are enjoyable or amusing
['I do it for the fun of it', 'he is fun to have around']
Lemma('fun.n.01.fun')
Lemma('fun.n.01.merriment')
Lemma('fun.n.01.playfulness')


fun.n.02
verbal wit or mockery (often at another's expense but not to be taken seriously)
['he became a figure of fun', 'he said it in sport']
Lemma('fun.n.02.fun')
Lemma('fun.n.02.play')
Lemma('fun.n.02.sport')


fun.n.03
violent and excited activity
['she asked for money and then the fun began', 'they began to fight like fun']
Lemma('fun.n.03.fun')


playfulness.n.02
a disposition to find (or make) causes for amusement
['her playfulness surprised me', 'he was fun to be with']
Lemma('playfulness.n.02.playfulness')
Lemma('playfulness.n.02.fun')
```

In [41]:
```
#How many differnt meanings :
for words in wordnet.synsets("Fun"):
    for lemma in words.lemmas():
        print(lemma)
    print("\n")
```
```
Lemma('fun.n.01.fun')
Lemma('fun.n.01.merriment')
Lemma('fun.n.01.playfulness')


Lemma('fun.n.02.fun')
Lemma('fun.n.02.play')
Lemma('fun.n.02.sport')


Lemma('fun.n.03.fun')


Lemma('playfulness.n.02.playfulness')
Lemma('playfulness.n.02.fun')
```

In [42]:
```
word = wordnet.synsets("Play")[0]

#Checking name :
print(word.name())

#Checking definition :
print(word.definition())

#Checking examples:
print(word.examples())
```
```
play.n.01
a dramatic work intended for performance by actors on a stage
['he wrote several plays but only one was produced on Broadway']
```

In [43]:
```
word = wordnet.synsets("Play")[0]

#Find more abstract term :
print(word.hypernyms())
```
```
[Synset('dramatic_composition.n.01')]
```

In [44]:
```
word = wordnet.synsets("Play")[0]

#Find more specific term :
word.hyponyms()
```
Out[44]:
```
[Synset('grand_guignol.n.01'),
 Synset('miracle_play.n.01'),
 Synset('morality_play.n.01'),
 Synset('mystery_play.n.01'),
 Synset('passion_play.n.01'),
 Synset('playlet.n.01'),
 Synset('satyr_play.n.01'),
 Synset('theater_of_the_absurd.n.01')]
```

In [45]:
```
word = wordnet.synsets("Play")[0]

#Get only name :
print(word.lemmas()[0].name())
```
```
play
```

In [46]:
```
#Finding synonyms :

#Empty list to store synonyms :
synonyms = []

for words in wordnet.synsets('Fun'):
    for lemma in words.lemmas():
        synonyms.append(lemma.name())

synonyms
```
Out[46]:
```
['fun',
 'merriment',
 'playfulness',
 'fun',
 'play',
 'sport',
 'fun',
 'playfulness',
 'fun']
```

In [47]:
```
#Finding antonyms :

#Empty list to store antonyms :
antonyms = []

for words in wordnet.synsets('Natural'):
    for lemma in words.lemmas():
        if lemma.antonyms():
            antonyms.append(lemma.antonyms()[0].name())

#Print antonyms :
antonyms
```
Out[47]:
```
['unnatural', 'artificial', 'supernatural', 'sharp']
```

In [48]:
```python
#Finding synonyms and antonyms :

#Empty lists to store synonyms/antonynms :
synonyms = []
antonyms = []

for words in wordnet.synsets('New'):
    for lemma in words.lemmas():
        synonyms.append(lemma.name())
        if lemma.antonyms():
            antonyms.append(lemma.antonyms()[0].name())

#Print lists :
print(synonyms)
print("\n")
print(antonyms)
```

```
['new', 'fresh', 'new', 'novel', 'raw', 'new', 'new', 'unexampled', 'new', 'new', 'newfangled', 'new', 'New',
'Modern', 'New', 'new', 'young', 'new', 'newly', 'freshly', 'fresh', 'new']


['old', 'worn']
```

In [49]:
```python
#Similarity in words :
word1 = wordnet.synsets("ship","n")[0]

word2 = wordnet.synsets("boat","n")[0]

#Check similarity :
print(word1.wup_similarity(word2))
```

```
0.9090909090909091
```

In [50]:
```python
#Similarity in words :
word1 = wordnet.synsets("ship","n")[0]

word2 = wordnet.synsets("bike","n")[0]

#Check similarity :
print(word1.wup_similarity(word2))
```

```
0.6956521739130435
```

In [51]:
```python
#Import required libraries :
from sklearn.feature_extraction.text import CountVectorizer

#Text for analysis :
sentences = ["Jim and Pam travelled by the bus:",
             "The train was late",
             "The flight was full.Travelling by flight is expensive"]

#Create an object :
cv = CountVectorizer()

cv
```

Out[51]:
```
CountVectorizer()
```

In [52]:
```python
#Generating output for Bag of Words :
B_O_W = cv.fit_transform(sentences).toarray()

#Total words with their index in model :
print(cv.vocabulary_)
print("\n")

#Features :
print(cv.get_feature_names())
print("\n")

#Show the output :
print(B_O_W)
```

```
{'jim': 7, 'and': 0, 'pam': 9, 'travelled': 12, 'by': 2, 'the': 10, 'bus': 1, 'train': 11, 'was': 14, 'late':
8, 'flight': 4, 'full': 5, 'travelling': 13, 'is': 6, 'expensive': 3}


['and', 'bus', 'by', 'expensive', 'flight', 'full', 'is', 'jim', 'late', 'pam', 'the', 'train', 'travelled', 't
ravelling', 'was']


[[1 1 1 0 0 0 0 1 0 1 1 0 1 0 0]
 [0 0 0 0 0 0 0 0 1 0 1 1 0 0 1]
 [0 0 1 1 2 1 1 0 0 0 1 0 0 1 1]]
```

In [53]:
```python
#Import required libraries :
from sklearn.feature_extraction.text import TfidfVectorizer

#Sentences for analysis :
sentences = ['This is the first document','This document is the second document']

#Create an object :
vectorizer = TfidfVectorizer(norm = None)

#Generating output for TF_IDF :
X = vectorizer.fit_transform(sentences).toarray()
```

In [54]:
```python
#Total words with their index in model :
print(vectorizer.vocabulary_)
```

```
{'this': 5, 'is': 2, 'the': 4, 'first': 1, 'document': 0, 'second': 3}
```

In [55]:
```python
#Features :
print(vectorizer.get_feature_names())
```

```
['document', 'first', 'is', 'second', 'the', 'this']
```

In [57]:
```python
X
```

Out[57]:
```
array([[1.        , 1.40546511, 1.        , 0.        , 1.        ,
        1.        ],
       [2.        , 0.        , 1.        , 1.40546511, 1.        ,
        1.        ]])
```

In [58]:
```python
import pandas as pd
```

View Feature Matrix As Data Frame :

In [60]:
```python
pd.DataFrame(X, columns=vectorizer.get_feature_names())
```

Out[60]:

	document	first	is	second	the	this
0	1.0	1.405465	1.0	0.000000	1.0	1.0
1	2.0	0.000000	1.0	1.405465	1.0	1.0

In [ ]: