Task 1: Getting Ready

In [1]:
```
# just to hide warnings
import warnings
warnings.filterwarnings('ignore')
```

In [9]:
```
pip install nltk
```

```
Requirement already satisfied: nltk in /home/chattha/anaconda3/lib/python3.11/site-packages (3.8.1)
Requirement already satisfied: click in /home/chattha/anaconda3/lib/python3.11/site-packages (from nltk)
(8.0.4)
Requirement already satisfied: joblib in /home/chattha/anaconda3/lib/python3.11/site-packages (from nltk)
(1.2.0)
Requirement already satisfied: regex>=2021.8.3 in /home/chattha/anaconda3/lib/python3.11/site-packages (f
rom nltk) (2022.7.9)
Requirement already satisfied: tqdm in /home/chattha/anaconda3/lib/python3.11/site-packages (from nltk)
(4.65.0)
Note: you may need to restart the kernel to use updated packages.
```

In [10]:
```
import nltk
```

In [11]:
```
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml (https://raw.githubuserc
ontent.com/nltk/nltk_data/gh-pages/index.xml)
```

Out[11]:
```
True
```

In [ ]:
```

```

In [12]:
```
from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

For Text 1

In [13]:
```
text1
```

Out[13]:
```
<Text: Moby Dick by Herman Melville 1851>
```

In [14]:
```
text1.concordance("Monster")
```

```
Displaying 25 of 49 matches:
des cometh within the chaos of this monster ' s mouth , be it beast , boat , or
nter into the dreadful gulf of this monster ' s ( whale ' s ) mouth , are immed
time with a lance ; but the furious monster at length rushed on the boat ; hims
 . Such a portentous and mysterious monster roused all my curiosity . Then the
and flank with the most exasperated monster . Long usage had , for this Stubb ,
ACK ).-- Under this head I reckon a monster which , by the various names of Fin
arned the history of that murderous monster against whom I and all the others h
ocity , cunning , and malice in the monster attacked ; therefore it was , that
iathan is restricted to the ignoble monster primitively pursued in the North ;
 and incontestable character of the monster to strike the imagination with unwo
mberment . Then , in darting at the monster , knife in hand , he had but given
e rock ; instead of this we saw the monster sailing off with the utmost gravity
e at Constantinople , a great sea - monster was captured in the neighboring Pro
 Of what precise species this sea - monster was , is not mentioned . But as he
man reasoning , Procopius ' s sea - monster , that for half a century stove the
hale ," as he called the fictitious monster which he declared to be incessantly
d his intention to hunt that mortal monster in person . But such a supposition
ng us on and on , in order that the monster might turn round upon us , and rend
d famous , and most deadly immortal monster , Don ;-- but that would be too lon
oluntarily lifted his voice for the monster , though for some little time past
s rescuing Andromeda from the sea - monster or whale . Where did Guido get the
 huge corpulence of that Hogarthian monster undulates on the surface , scarcely
nd is drawn just balancing upon the monster ' s spine ; and standing in that pr
 of cutting - in ) hove over to the monster as if to a quay ; and a boat , hurr
eet in length . They fancy that the monster to which these arms belonged ordina
```

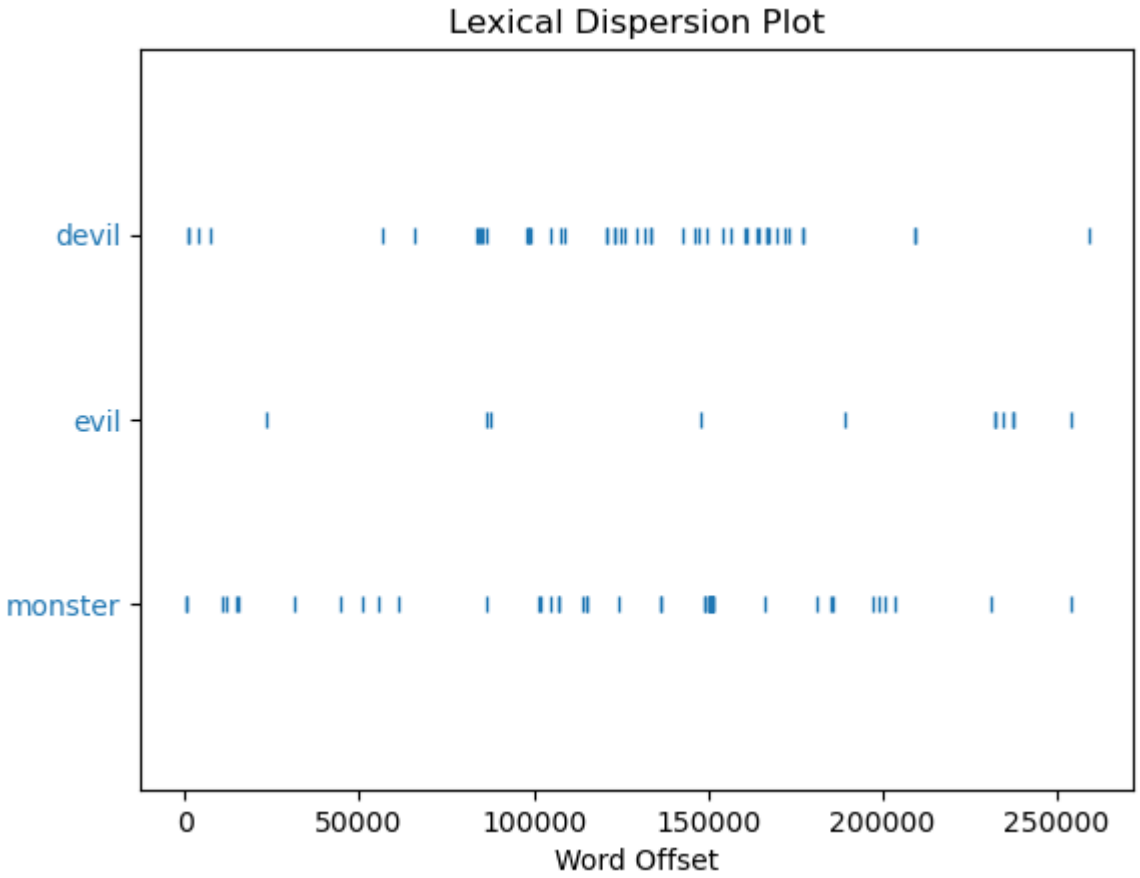
In [15]:
```
text1.similar("monster")
```

```
whale ship world sea whales boat pequod other sun leviathan thing king
water head captain air crew cabin body more
```

In [16]:
```
text1.common_contexts(["monster","person"])
```

```
the_that
```

In [17]:
```
text1.dispersion_plot(["monster","evil","devil"])
```



In [18]:
```
set(text1)
```

```
 'mocked',
 'captured',
 'vernal',
 'glided',
 'instrument',
 'sleights',
 'said',
 'yourselves',
 'isn',
 'happy',
 'resurrection',
 'delude',
 'unsay',
 'wigwam',
 'lengthen',
 'uncounted',
 'landsmen',
 'Presbyterian',
 'workmen',
 'Excellent',
```

Text-1

In [66]:
```
print(text1)
print(len(text1))
print(len(set(text1)))
lexical_richness_text1 = len(set(text1))/len(text1)
print(lexical_richness_text1)
```

```
<Text: Moby Dick by Herman Melville 1851>
260819
19317
0.07406285585022564
```

In [67]:
```
set(text1)
```

```
 'MIRABILIS',
 'competent',
 'toe',
 'wrangling',
 'uttons',
 'observable',
 'Suspended',
 'obliquity',
 'grappled',
 'magnificence',
 'layn',
 'TWISTED',
 '31',
 'velvet',
 'protection',
 'foamy',
 'controlling',
 'Forehead',
 'peacefulness',
 'Killed',
```

Text-2

In [23]:
```
print(text2)
print(len(text2))
print(len(set(text2)))
lexical_richness_text2 = len(set(text2))/len(text2)
print(lexical_richness_text2)
```

```
<Text: Sense and Sensibility by Jane Austen 1811>
141576
6833
0.04826383002768831
```

In [24]:
```
set(text2)
```

```
 'sourness',
 'annihilation',
 'joys',
 'rung',
 'awoke',
 'Writing',
 'brush',
 'appetites',
 'apologies',
 'introducing',
 'occasion',
 'convincing',
 'surpassed',
 'preferment',
 'dreadful',
 'Could',
 'lodges',
 'detested',
 'Exert',
 'gratitude'
```

Text-3

In [26]:
```
print(text3)
print(len(text3))
print(len(set(text3)))
lexical_richness_text3 = len(set(text3))/len(text3)
print(lexical_richness_text3)
```

```
<Text: The Book of Genesis>
44764
2789
0.06230453042623537
```

In [27]:
```
set(text3)
```

Out[27]:
```
{'ways',
 'household',
 'fath',
 'beneath',
 'desired',
 'draw',
 'embalm',
 'mourning',
 '.',
 'wilderness',
 'wrought',
 'blameless',
 'covenant',
 'sin',
 'told',
 'neither',
 'mocked',
 'all',
 'ewe',
```

Text-4

In [29]:
```
print(text4)
print(len(text4))
print(len(set(text4)))
lexical_richness_text4 = len(set(text4))/len(text4)
print(lexical_richness_text4)
```

```
<Text: Inaugural Address Corpus>
152901
10025
0.06556530042314962
```

In [30]:
```
set(text4)
```

Out[30]:
```
{'deficits',
 'exterior',
 'disastrous',
 'commands',
 'Congressman',
 '.',
 'regarding',
 'competent',
 'bipartisanship',
 'observable',
 'magnificence',
 'fiat',
 'impracticable',
 'protection',
 'controlling',
 'Information',
 'detriment',
 'capitol',
 'Greater',
```

Text 5

In [32]:
```
print(text5)
print(len(text5))
print(len(set(text5)))
lexical_richness_text5 = len(set(text5))/len(text5)
print(lexical_richness_text5)
```

```
<Text: Chat Corpus>
45010
6066
0.13477005109975562
```

In [33]:
```
set(text5)
```

Out[33]:
```
{'',
 'EST',
 'pussies',
 'ciggareets',
 'shhhh',
 '.',
 'sayn',
 'regarding',
 'toe',
 '--------------------',
 'bitdh',
 '31',
 'controlling',
 'U542',
 'all',
 'CALI',
 'said',
 'happy',
 'chanop',
```

Text 5

In [35]:
```
print(text5)
print(len(text5))
print(len(set(text5)))
lexical_richness_text5 = len(set(text5))/len(text5)
print(lexical_richness_text5)
```

```
<Text: Chat Corpus>
45010
6066
0.13477005109975562
```

In [36]:
```
set(text5)
```

Out[36]:
```
{'',
 'EST',
 'pussies',
 'ciggareets',
 'shhhh',
 '.',
 'sayn',
 'regarding',
 'toe',
 '--------------------',
 'bitdh',
 '31',
 'controlling',
 'U542',
 'all',
 'CALI',
 'said',
 'happy',
 'chanop',
```

Text-6

In [38]:
```
print(text6)
print(len(text6))
print(len(set(text6)))
lexical_richness_text6 = len(set(text6))/len(text6)
print(lexical_richness_text6)
```

```
<Text: Monty Python and the Holy Grail>
16967
2166
0.1276595744680851
```

In [39]:
```
set(text6)
```

Out[39]:
```
{'Bristol',
 'sacred',
 'ways',
 'draw',
 'commands',
 'Pure',
 '.',
 '10',
 'temptress',
 'question',
 '#',
 'easily',
 'coconut',
 'told',
 'eh',
 'worst',
 'all',
 'said',
 'strength',
 'happy'
```

Text-7

In [41]:
```
print(text7)
print(len(text7))
print(len(set(text7)))
lexical_richness_text7 = len(set(text7))/len(text7)
print(lexical_richness_text7)
```

```
<Text: Wall Street Journal>
100676
12408
0.12324685128531129
```

In [42]:
```
set(text7)
```

```
 'saving',
 'joys',
 'slowdowns',
 'outlets',
 'rung',
 'orange',
 'hierarchical',
 'Intelogic',
 'promptly',
 'furor',
 'quantitive',
 'introducing',
 'doctorate',
 '*-106',
 '20-point',
 'Hoosier',
 'chronicle',
 '0.4',
 'dreadful',
```

Text-8

In [44]:
```
print(text8)
print(len(text8))
print(len(set(text8)))
lexical_richness_text8 = len(set(text8))/len(text8)
print(lexical_richness_text8)
```

```
<Text: Personals Corpus>
4867
1108
0.22765564002465585
```

In [45]:
```
set(text8)
```

Out[45]:
```
{'older',
 'ties',
 'mature',
 '37yrs',
 'SOH',
 'Charters',
 'missing',
 'Knox',
 'fship',
 '36',
 'Suburbs',
 'Female',
 'always',
 'Gent',
 '.',
 '10',
 'trustworthy',
 'cut',
 'MID',
```

Text-9

In [47]:
```
print(text9)
print(len(text9))
print(len(set(text9)))
lexical_richness_text9 = len(set(text9))/len(text9)
print(lexical_richness_text9)
```

```
<Text: The Man Who Was Thursday by G . K . Chesterton 1908>
69213
6807
0.0983485761345412
```

In [48]:
```
set(text9)
```

Out[48]:
```
{'exterior',
 'madder',
 'Pantheist',
 'fanatics',
 'commands',
 'disastrous',
 '.',
 'trot',
 'blameless',
 'velvet',
 'hairbreadth',
 'all',
 'captured',
 'instrument',
 'said',
 'abnormally',
 'yourselves',
 'happy',
 'isn',
```

In [49]:
```
text = [text1, text2, text3, text4, text5, text6, text7, text8, text9]
print("="*10,"Names of Texts Used", 10 *"=", "\n")
for idx, i in enumerate(text, start = 1):
    print(f"Name of Text {idx} is ----->> ", i)

print("\n","="*10, "Length of Texts Used","="*10,"\n")

for idx, i in enumerate(text, start = 1):
    print(f"Length of Text {idx} is ----->> ",len(i))

print("\n","="*10, "Unique Words of Texts Used","="*10,"\n")
for idx, i in enumerate(text, start = 1):
    print(f"Length of Unique words in Text {idx} ----->> ",len(set(i)))
    #print(set(i))

print("\n","="*10, "Lexical Richness of Texts Used","="*10,"\n")
for idx, i in enumerate(text, start = 1):
    lexical_richness_text = len(set(i))/len(i)
    print(f"Lexical Richness of Text {idx} ----->>",lexical_richness_text)
```

```
========== Names of Texts Used ==========

Name of Text 1 is ----->>  <Text: Moby Dick by Herman Melville 1851>
Name of Text 2 is ----->>  <Text: Sense and Sensibility by Jane Austen 1811>
Name of Text 3 is ----->>  <Text: The Book of Genesis>
Name of Text 4 is ----->>  <Text: Inaugural Address Corpus>
Name of Text 5 is ----->>  <Text: Chat Corpus>
Name of Text 6 is ----->>  <Text: Monty Python and the Holy Grail>
Name of Text 7 is ----->>  <Text: Wall Street Journal>
Name of Text 8 is ----->>  <Text: Personals Corpus>
Name of Text 9 is ----->>  <Text: The Man Who Was Thursday by G . K . Chesterton 1908>

 ========== Length of Texts Used ==========

Length of Text 1 is ----->>  260819
Length of Text 2 is ----->>  141576
Length of Text 3 is ----->>  44764
Length of Text 4 is ----->>  152901
Length of Text 5 is ----->>  45010
Length of Text 6 is ----->>  16967
Length of Text 7 is ----->>  100676
Length of Text 8 is ----->>  4867
Length of Text 9 is ----->>  69213

 ========== Unique Words of Texts Used ==========

Length of Unique words in Text 1 ----->>  19317
Length of Unique words in Text 2 ----->>  6833
Length of Unique words in Text 3 ----->>  2789
Length of Unique words in Text 4 ----->>  10025
Length of Unique words in Text 5 ----->>  6066
Length of Unique words in Text 6 ----->>  2166
Length of Unique words in Text 7 ----->>  12408
Length of Unique words in Text 8 ----->>  1108
Length of Unique words in Text 9 ----->>  6807

 ========== Lexical Richness of Texts Used ==========

Lexical Richness of Text 1 ----->> 0.07406285585022564
Lexical Richness of Text 2 ----->> 0.04826383002768831
Lexical Richness of Text 3 ----->> 0.06230453042623537
```

```
Lexical Richness of Text 4 ----->> 0.06556530042314962
Lexical Richness of Text 5 ----->> 0.13477005109975562
Lexical Richness of Text 6 ----->> 0.1276595744680851
Lexical Richness of Text 7 ----->> 0.12324685128531129
Lexical Richness of Text 8 ----->> 0.22765564002465585
Lexical Richness of Text 9 ----->> 0.0983485761345412
```

Task 2: Term Frequency

In [50]:
```
def TF(word,corpus):
    tf = (text1.count(word) / len(corpus)) * 100
    return tf
```

In [51]:
```
print(TF("monster",text1))
```

```
0.018786974875296663
```

In [52]:
```
import math
def TFLOG(word,corpus):
    return math.log(corpus.count(word)+1,10)
```

In [53]:
```
print(TFLOG(".",text1))
```

```
3.836513998890671
```

In [54]:
```
def IDF(word,corpus):
    return math.log(9 / corpus.count(word), 10)
```

In [55]:
```
print(IDF(".",text1))
```

```
-2.8822082042808295
```

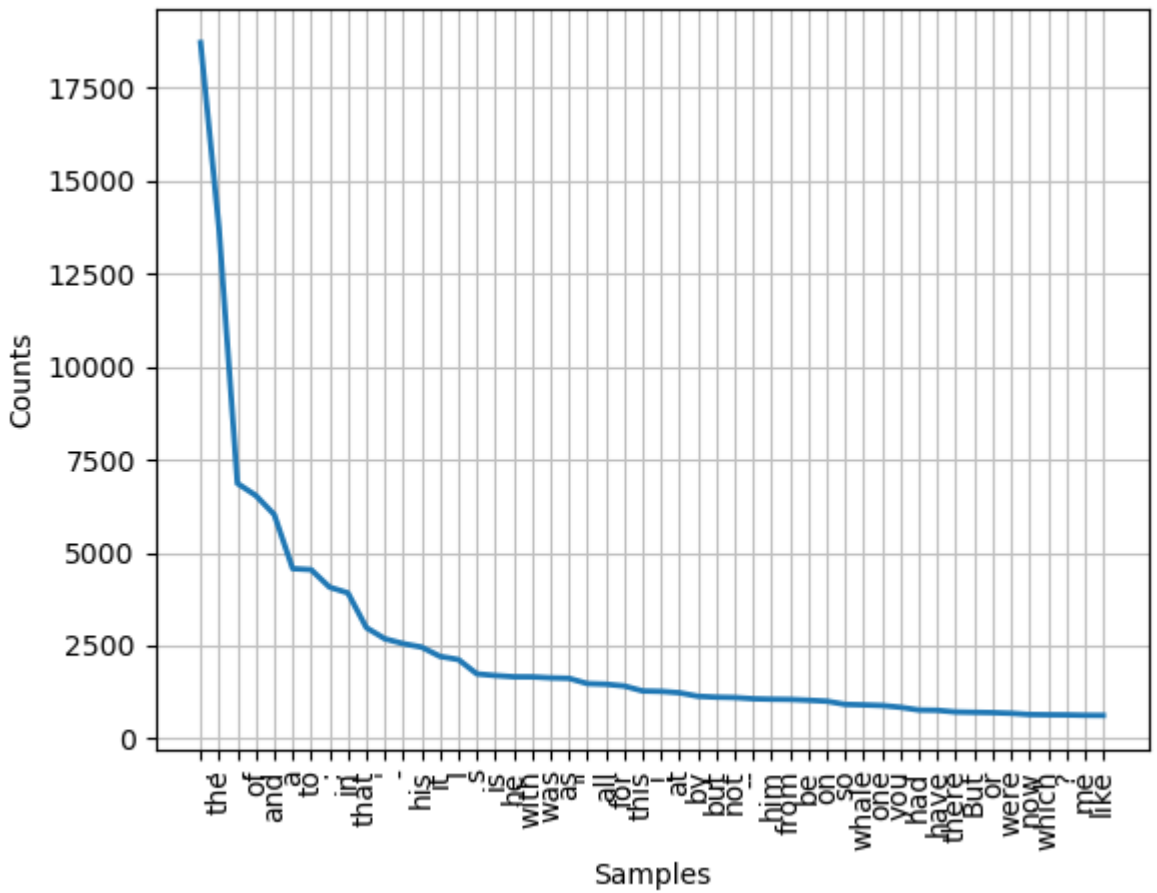
In [56]:
```
freqdist = FreqDist(text1)
freqdist.most_common(10)
```

Out[56]:
```
[(',', 18713),
 ('the', 13721),
 ('.', 6862),
 ('of', 6536),
 ('and', 6024),
 ('a', 4569),
 ('to', 4542),
 (';', 4072),
 ('in', 3916),
 ('that', 2982)]
```

In [ ]:
```

```

In [57]:
```
freqdist.plot(50)
```



Out[57]: <Axes: xlabel='Samples', ylabel='Counts'>

In [58]:
```
freq = FreqDist(text1)
```

In [59]:
```
freq.most_common(3)
```

Out[59]: [(',', 18713), ('the', 13721), ('.', 6862)]

In [ ]:

In [60]:
```
import math
def TF(word,text):
    tf = (text.count(word) / len(text)) * 100
    return tf

def TFLOG(word,text):
    return math.log(text.count(word)+1,10)

def IDF(word,text):
    return math.log(9 / text.count(word), 10)


words = ["monster", "evil", "devil", "the"]
for i in words:
    print("\nStart for word ----->> ", i, "\n")
    print(f'Term Frequency of "{i}" ----->> ', TF(i, text1))
    print(f'Term Frequency Log "{i}" ----->> ', TFLOG(i,text1))
    print(f'Inverse Document Frequency "{i}" ----->> ',IDF(i,text1))
    print("="*50)

print("="*50)



most_freq = FreqDist(text1)
freq = most_freq.most_common(3)
for value, _ in freq:
    print("\nStart for Most Common word ----->>",value, "\n")
    print(f'Term Frequency of "{value}" ----->> ', TF(value, text1))
    print(f'Term Frequency Log of"{value}" ----->> ', TFLOG(value,text1))
    print(f'Inverse Document Frequency of "{value}" ----->> ', IDF(value,text1))
    print("="*50)
```

```
Start for word ----->>  monster

Term Frequency of "monster" ----->>  0.018786974875296663
Term Frequency Log "monster" ----->>  1.6989700043360185
Inverse Document Frequency "monster" ----->>  -0.7359535705891886
==================================================

Start for word ----->>  evil

Term Frequency of "evil" ----->>  0.004217484155678842
Term Frequency Log "evil" ----->>  1.0791812460476247
Inverse Document Frequency "evil" ----->>  -0.08715017571890013
==================================================

Start for word ----->>  devil

Term Frequency of "devil" ----->>  0.01955379017632918
Term Frequency Log "devil" ----->>  1.716003343634799
Inverse Document Frequency "devil" ----->>  -0.7533276666586114
==================================================

Start for word ----->>  the

Term Frequency of "the" ----->>  5.260736372733581
Term Frequency Log "the" ----->>  4.137417414990392
Inverse Document Frequency "the" ----->>  -3.1831432548946452
==================================================
==================================================

Start for Most Common word ----->> ,

Term Frequency of "," ----->>  7.174707364110744
Term Frequency Log of"," ----->>  4.272166625140787
Inverse Document Frequency of "," ----->>  -3.3179009081517252
==================================================

Start for Most Common word ----->> the

Term Frequency of "the" ----->>  5.260736372733581
Term Frequency Log of"the" ----->>  4.137417414990392
```

```
Inverse Document Frequency of "the" ----->>  -3.1831432548946452
==================================================

Start for Most Common word ----->> .

Term Frequency of "." ----->>  2.630943297842565
Term Frequency Log of"." ----->>  3.836513998890671
Inverse Document Frequency of "." ----->>  -2.8822082042808295
==================================================
```

Task 3: Tokenization & POS

In [61]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /home/chattha/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[61]:
```
True
```

In [62]:
```
text = "NLTK is a powerful library for natural language processing."
words = nltk.word_tokenize(text)
sentences = nltk.sent_tokenize(text)
print(words)
print(sentences)
```

```
['NLTK', 'is', 'a', 'powerful', 'library', 'for', 'natural', 'language', 'processing', '.']
['NLTK is a powerful library for natural language processing.']
```

In [63]:
```
tags = nltk.pos_tag(words)
print(tags)
```

```
[('NLTK', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('powerful', 'JJ'), ('library', 'NN'), ('for', 'IN'), ('nat
ural', 'JJ'), ('language', 'NN'), ('processing', 'NN'), ('.', '.')]
```

In [64]:
```
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/chattha/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[64]:
```
True
```

In [65]:
```
from nltk.corpus import stopwords
```

In [313]:
```
filtered_words = [word for word in words if word.lower() not in stopwords.words('english')]
print(filtered_words)
```

```
['NLTK', 'powerful', 'library', 'natural', 'language', 'processing', '.']
```

In [ ]:
```

```