JawadAhmed_20P-0165_LAB01

December 3, 2023

1 Lab01: Getting Started with NLTK

1.1 Task1: Getting Ready

[1]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in /home/jadi/anaconda3/lib/python3.7/site-
packages (3.4.5)
Requirement already satisfied: six in /home/jadi/anaconda3/lib/python3.7/site-
packages (from nltk) (1.14.0)
DEPRECATION: pyodbc 4.0.0-unsupported has a non-standard version number.
pip 24.0 will enforce this behaviour change. A possible replacement is to
upgrade to a newer version of pyodbc or contact the author to suggest that they
release a version with a conforming version number. Discussion can be found at
https://github.com/pypa/pip/issues/12063
```

[2]:
```
import nltk
nltk.download()
```

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

[2]:
```
True
```

[3]:
```
# Download the book module "typically refer to the collection of texts"
nltk.download('book')
```

```
[nltk_data] Downloading collection 'book'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /home/jadi/nltk_data…
[nltk_data]    |   Package abc is already up-to-date!
[nltk_data]    | Downloading package brown to /home/jadi/nltk_data…
[nltk_data]    |   Package brown is already up-to-date!
[nltk_data]    | Downloading package chat80 to /home/jadi/nltk_data…
[nltk_data]    |   Package chat80 is already up-to-date!
[nltk_data]    | Downloading package cmudict to
```

```
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package conll2000 to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package conll2000 is already up-to-date!
[nltk_data]    | Downloading package conll2002 to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package conll2002 is already up-to-date!
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package dependency_treebank is already up-to-date!
[nltk_data]    | Downloading package genesis to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package ieer to /home/jadi/nltk_data…
[nltk_data]    |   Package ieer is already up-to-date!
[nltk_data]    | Downloading package inaugural to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package inaugural is already up-to-date!
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package nps_chat to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package nps_chat is already up-to-date!
[nltk_data]    | Downloading package names to /home/jadi/nltk_data…
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package ppattach to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package ppattach is already up-to-date!
[nltk_data]    | Downloading package reuters to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package reuters is already up-to-date!
[nltk_data]    | Downloading package senseval to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package senseval is already up-to-date!
[nltk_data]    | Downloading package state_union to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package state_union is already up-to-date!
[nltk_data]    | Downloading package stopwords to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package swadesh to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package swadesh is already up-to-date!
```

```
[nltk_data]    | Downloading package timit to /home/jadi/nltk_data…
[nltk_data]    |   Package timit is already up-to-date!
[nltk_data]    | Downloading package treebank to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package toolbox to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package toolbox is already up-to-date!
[nltk_data]    | Downloading package udhr to /home/jadi/nltk_data…
[nltk_data]    |   Package udhr is already up-to-date!
[nltk_data]    | Downloading package udhr2 to /home/jadi/nltk_data…
[nltk_data]    |   Package udhr2 is already up-to-date!
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package unicode_samples is already up-to-date!
[nltk_data]    | Downloading package webtext to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package webtext is already up-to-date!
[nltk_data]    | Downloading package wordnet to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to /home/jadi/nltk_data…
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package maxent_treebank_pos_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package universal_tagset is already up-to-date!
[nltk_data]    | Downloading package punkt to /home/jadi/nltk_data…
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package book_grammars is already up-to-date!
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package city_database is already up-to-date!
[nltk_data]    | Downloading package tagsets to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package tagsets is already up-to-date!
[nltk_data]    | Downloading package panlex_swadesh to
```

```
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package panlex_swadesh is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /home/jadi/nltk_data…
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection book
```

[3]:
```
True
```

[4]:
```
# import all the datasets and functions provided by book module
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
```

[5]:
```
# Print all the occurances of the 'monster' word along with some surronding␣
 ↪context (some words)
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
```

[6]:
```
# Return the similar words like 'monster' that are used in similar context as␣
 ↪monster
text1.similar("monster")
```

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

[7]:
```
text1.common_contexts(["monster", "person"])
```

```
the_that
```

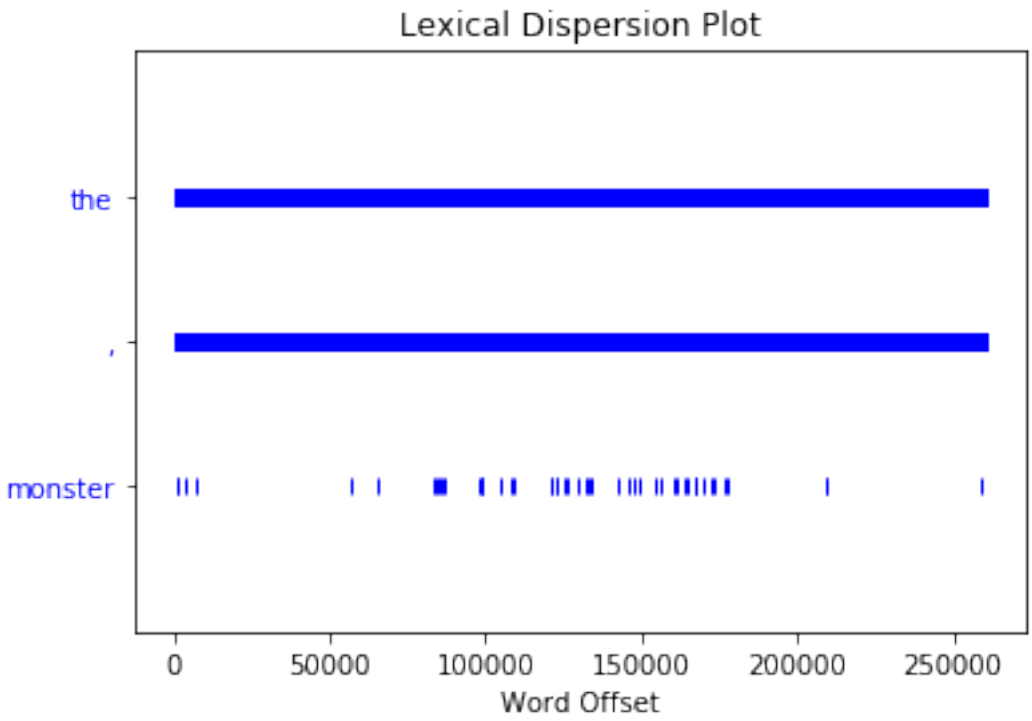
[8]:
```
text1.common_contexts(["might", "turn"])
```

```
No common contexts were found
```

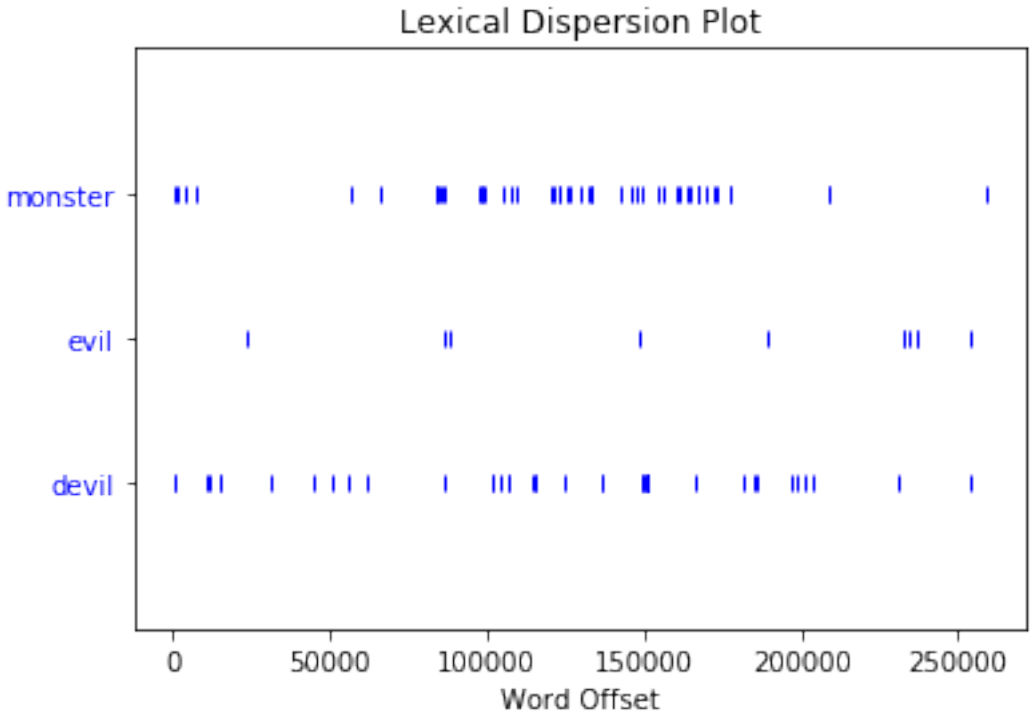
[9]:
```
text1.common_contexts(['monster', 'mouth'])
```

```
the_and
```

[10]:
```
# Dispersion plot show the relative position of the words in the given text,␣
 ↪where they occur and how they are distributed in the text
text1.dispersion_plot(["the", ',', 'monster'])
```



[11]:
```
text1.dispersion_plot(["monster", 'evil', 'devil'])
```



[12]:
```
# No repetition allowed in set, set to get all the unique words
# Printing only 10 as a lot of unique words
unique_words = list(set(text1))
print(len(unique_words))
print(unique_words[:10])
```

```
19317
['territories', 'Counterpane', 'She', 'idol', 'tragic', 'stages', 'sick',
'fashions', 'shriek', 'prophet']
```

[13]:
```
# View the sorted list
sorted_text1_unique_words = sorted(set(text1))
```

[14]:
```
print("First 10: ", sorted_text1_unique_words[:10])
print("Last 10: ", sorted_text1_unique_words[-10:])
```

```
First 10:  ['!', '!"', '!"--', "!'", '!\'"', '!)', '!)"', '!*', '!--', '!--"']
Last 10:  ['zag', 'zay', 'zeal', 'zephyr', 'zig', 'zodiac', 'zone', 'zoned',
'zones', 'zoology']
```

[15]:
```
# Check the sizes of the vocabulary for different texts
vocab_size_text1 = len(text1)
vocab_size_text2 = len(text2)
vocab_size_text3 = len(text3)
vocab_size_text4 = len(text4)
vocab_size_text5 = len(text5)
vocab_size_text6 = len(text6)
vocab_size_text7 = len(text7)
vocab_size_text8 = len(text8)
vocab_size_text9 = len(text9)

print(f"Vocabulary size for text1: {vocab_size_text1}")
print(f"Vocabulary size for text2: {vocab_size_text2}")
print(f"Vocabulary size for text3: {vocab_size_text3}")
print(f"Vocabulary size for text4: {vocab_size_text4}")
print(f"Vocabulary size for text5: {vocab_size_text5}")
print(f"Vocabulary size for text6: {vocab_size_text6}")
print(f"Vocabulary size for text7: {vocab_size_text7}")
print(f"Vocabulary size for text8: {vocab_size_text8}")
print(f"Vocabulary size for text9: {vocab_size_text9}")
```

```
Vocabulary size for text1: 260819
Vocabulary size for text2: 141576
Vocabulary size for text3: 44764
Vocabulary size for text4: 152901
Vocabulary size for text5: 45010
```

```
Vocabulary size for text6: 16967
Vocabulary size for text7: 100676
Vocabulary size for text8: 4867
Vocabulary size for text9: 69213
```

1.1.1 Lexical Richness

Lexical richness is a measure of the diversity and variety of words used in a piece of text.

$$\text{Lexical Richness} = \frac{\text{Number of Unique Words}}{\text{Total Number of Words}}$$

[16]:
```
!pip install latexify-py==0.2.0
```

```
Requirement already satisfied: latexify-py==0.2.0 in
/home/jadi/anaconda3/lib/python3.7/site-packages (0.2.0)
Requirement already satisfied: dill>=0.3.2 in
/home/jadi/anaconda3/lib/python3.7/site-packages (from latexify-py==0.2.0)
(0.3.7)
DEPRECATION: pyodbc 4.0.0-unsupported has a non-standard version number.
pip 24.0 will enforce this behaviour change. A possible replacement is to
upgrade to a newer version of pyodbc or contact the author to suggest that they
release a version with a conforming version number. Discussion can be found at
https://github.com/pypa/pip/issues/12063
```

[17]:
```
import latexify

@latexify.function
def lexical_richness(text):
    unique_words = set(text)
    return len(unique_words) / len(text)
```

[18]:
```
lexical_richness
```

[18]:

$$\begin{aligned} unique_words &= \text{set}\,(text) \\ \text{lexical}_\text{r}\text{ichness}(text) &= \frac{\text{len}(unique_words)}{\text{len}(text)} \end{aligned}$$

[19]:
```
# To get the corpous name
print(text1.name)
```

```
Moby Dick by Herman Melville 1851
```

[20]:
```
# To get the corpous length
print(len(text1))
```

```
260819
```

[21]:
```
# To get the unique words
print(len(set(text1)))
```

```
19317
```

[22]:
```
# To get the lexical richness of any corpous
lexical_richness_text1 = lexical_richness(text1)
print("Lexical richness of text1 => ", lexical_richness_text1)
```

```
Lexical richness of text1 =>  0.07406285585022564
```

1.2 Table

[23]:
```
# Function to fill the table
def fill_table(corpus_names, corpora):
    # Table header
    header = f"| {'Corpus': <15}"
    header += ' | '.join([f'{name: <15}' for name in corpus_names]) + ' |'
    print(header)

    # Table separator
    separator = f"| {'-' * 15}"
    separator += '|'.join(['-' * 15 for _ in corpus_names]) + '|'
    print(separator)

    # Table content
    for attribute in ["Corpus Name", "Corpus Length", "Unique Words", "Lexical␣
↪Richness"]:
        row = f"| {attribute: <15}"
        for corpus_name, corpus in zip(corpus_names, corpora):
            if attribute == "Corpus Name":
                row += f"| {getattr(corpus, 'name', ''): <15}"
            elif attribute == "Corpus Length":
                row += f"| {len(corpus): <15}"
            elif attribute == "Unique Words":
                row += f"| {len(set(corpus)): <15}"
            elif attribute == "Lexical Richness":
                row += f"| {lexical_richness(corpus): <15.4f}"
        print(row + "\n")

    # Table bottom line
    bottom_line = f"| {'-' * 15}"
    bottom_line += '|'.join(['-' * 15 for _ in corpus_names]) + '|'
```

```
    print(bottom_line)

# Example usage
corpus_names = ["Text1", "Text2", "Text3", "Text4", "Text5", "Text6", "Text7",␣
 ↪"Text8", "Text9"]
corpora = [text1, text2, text3, text4, text5, text6, text7, text8, text9]

fill_table(corpus_names, corpora)
```

```
| Corpus         Text1           | Text2           | Text3           | Text4
| Text5           | Text6           | Text7           | Text8           | Text9
|
| -----------------------------|---------------|---------------|---------------
|---------------|---------------|---------------|---------------|---------------
|
| Corpus Name    | Moby Dick by Herman Melville 1851| Sense and Sensibility by
Jane Austen 1811| The Book of Genesis| Inaugural Address Corpus| Chat Corpus
| Monty Python and the Holy Grail| Wall Street Journal| Personals Corpus| The
Man Who Was Thursday by G . K . Chesterton 1908

| Corpus Length  | 260819         | 141576         | 44764          | 152901
| 45010          | 16967          | 100676         | 4867           | 69213

| Unique Words   | 19317          | 6833           | 2789           | 10025
| 6066           | 2166           | 12408          | 1108           | 6807

| Lexical Richness| 0.0741         | 0.0483         | 0.0623         | 0.0656
| 0.1348         | 0.1277         | 0.1232         | 0.2277         | 0.0983

| -----------------------------|---------------|---------------|---------------
|---------------|---------------|---------------|---------------|---------------
|
```

2 Task 2: Term Frequency

[24]:
```
# Count of frequency of a particular word
text1.count('monster') / len(text1) * 100
```

[24]:
```
0.018786974875296663
```

[25]:
```
@latexify.function
def TF(corpus, token):
    token_count_in_corpus = corpus.count(token)
    total_terms_in_corpus = len(corpus)
    return token_count_in_corpus / total_terms_in_corpus * 100
```

[26]:
```
TF
```

[26]:

$$token_count_in_corpus = \text{corpus.count}\left(token\right)$$
$$total_terms_in_corpus = \text{len}\left(corpus\right)$$
$$\text{TF}(corpus, token) = \frac{token_count_in_corpus}{total_terms_in_corpus} 100$$

[27]:
```
TF(text1, 'monster')
```

[27]:
```
0.018786974875296663
```

[28]:
```
# Log Scales
import math

@latexify.function
def LOGTF(corpus, token):
    token_count_in_corpus = corpus.count(token)
    total_terms_in_corpus = len(corpus)

    return math.log(token_count_in_corpus + 1, 10)
```

[29]:
```
math.log(text1.count('monster')+1,10)
```

[29]:
```
1.6989700043360185
```

[30]:
```
LOGTF
```

[30]:

$$token_count_in_corpus = \text{corpus.count}\left(token\right)$$
$$total_terms_in_corpus = \text{len}\left(corpus\right)$$
$$\text{LOGTF}(corpus, token) = \log\left(token_count_in_corpus + 1, 10\right)$$

[31]:
```
LOGTF(text1, 'monster'), LOGTF(text1, ','), LOGTF(text1, 'the')
```

[31]:
```
(1.6989700043360185, 4.272166625140787, 4.137417414990392)
```

[32]:
```
# IDF: how often a term occured in the collection of documents

@latexify.function
def IDF(token, corpus, number_of_corpuses=9):
    count_of_token_in_one_corpus = corpus.count(token)

    return math.log(((number_of_corpuses + 1) / (count_of_token_in_one_corpus +␣
↪1)), 10)
```

[33]:
```
IDF
```

[33]:

$$count_of_token_in_one_corpus = \text{corpus.count}\,(token)$$
$$\text{IDF}(token, corpus, number_of_corpuses) = \log\left(\frac{number_of_corpuses+1}{count_of_token_in_one_corpus+1}, 10\right)$$

[34]:
```
IDF('monster', text1)
```

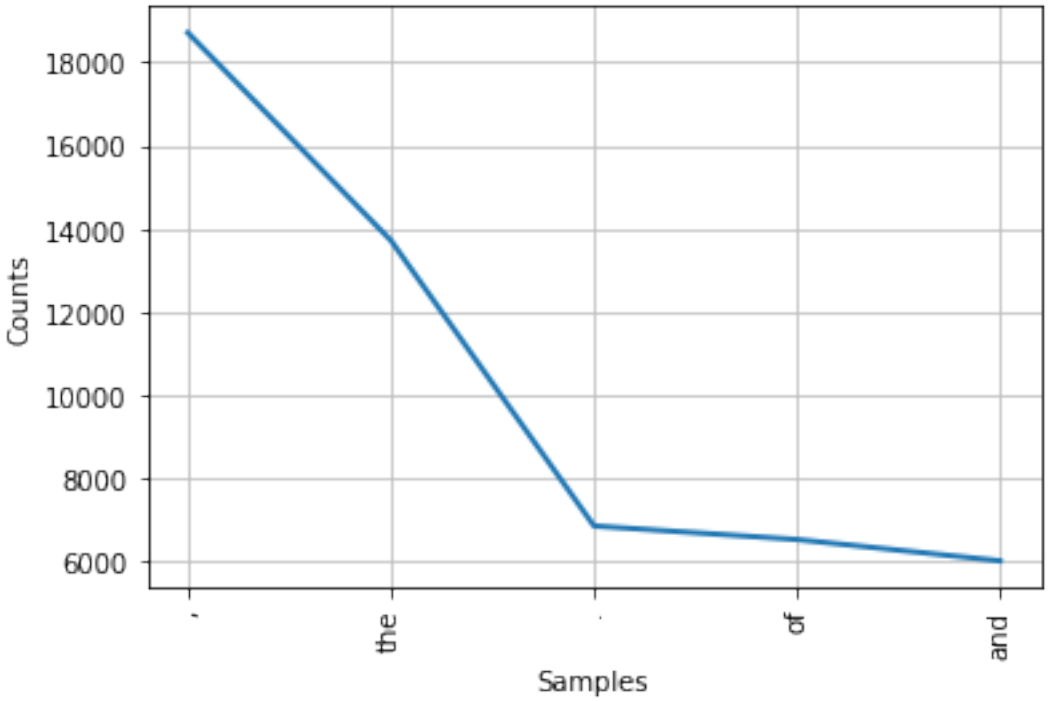
[34]:
```
-0.6989700043360187
```

[35]:
```
# Frequency Distribution
fdist1 = FreqDist(text1)
```

[36]:
```
fdist1.most_common(3)
```

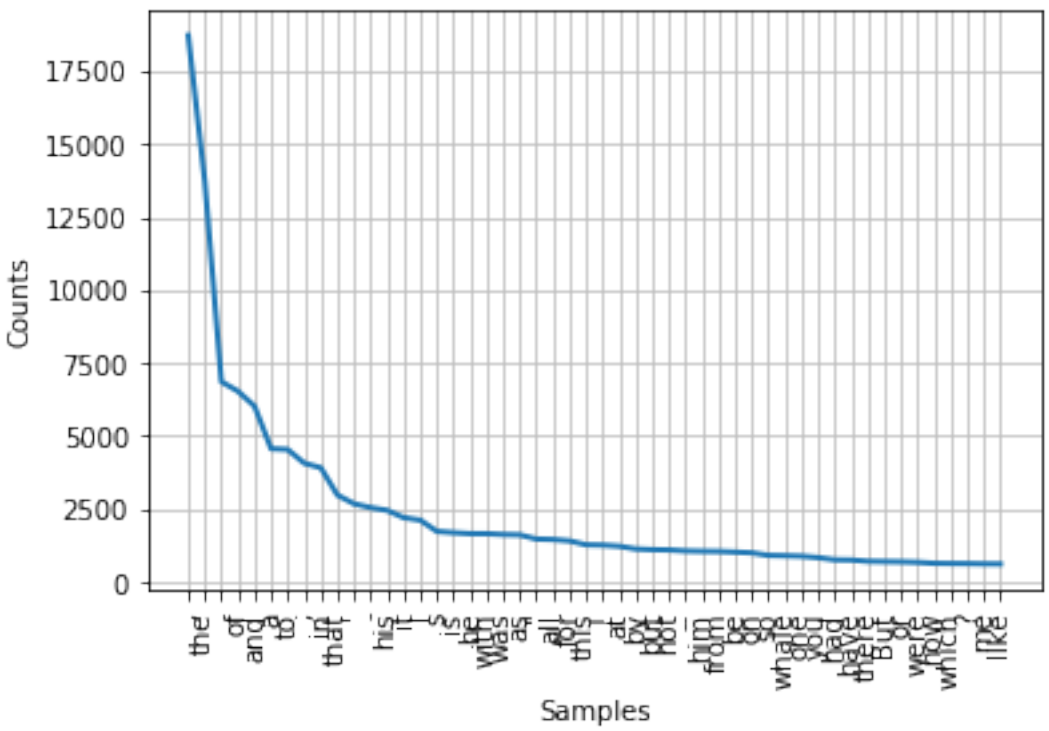
[36]:
```
[(',', 18713), ('the', 13721), ('.', 6862)]
```

[37]:
```
fdist1.plot(5)
```



[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efc576df9d0>
```

[38]:
```
fdist1.plot(50)
```



[38]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efc57638090>
```

[39]:
```
common_word_1 = fdist1.most_common(1)[0][0]
common_word_2 = fdist1.most_common(2)[1][0]
common_word_3 = fdist1.most_common(3)[2][0]
```

[40]:
```
print(common_word_1, common_word_2, common_word_3)
```

```
, the .
```

[41]:
```
# Define the tokens
tokens = ['monster', 'evil', 'devil', 'the', common_word_1, common_word_2,␣
 ↪common_word_3]

# Fill in the table for Text1
text1_values = [TF(text1, token) for token in tokens]
text1_log_values = [LOGTF(text1, token) for token in tokens]
text1_idf_values = [IDF(token, text1) for token in tokens]

# Print the table header
print(f"{'Token': <15} {'TF()': <15} {'LOGTF()': <15} {'IDF()': <15}")
print("-" * 60)
```

```
# Print the values for each token
for i in range(len(tokens)):
    token = tokens[i]
    tf_value = text1_values[i]
    logtf_value = text1_log_values[i]
    idf_value = text1_idf_values[i]

    print(f"{token: <15} {tf_value: <15.2f} {logtf_value: <15.2f} {idf_value:␣
↪<15.4f}")
```

```
Token           TF()            LOGTF()         IDF()
------------------------------------------------------------
monster         0.02            1.70            -0.6990
evil            0.00            1.08            -0.0792
devil           0.02            1.72            -0.7160
the             5.26            4.14            -3.1374
,               7.17            4.27            -3.2722
the             5.26            4.14            -3.1374
.               2.63            3.84            -2.8365
```

3 Task 3: Tokenization & POS

[42]:
```
# Tokenization of the sentences and words
nltk.download('punkt')
```

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

[42]:
```
True
```

[43]:
```
text = "NLTK is a powerful library for natural language processing."
words = nltk.word_tokenize(text)
sentences = nltk.sent_tokenize(text)
```

[44]:
```
print(words)
print(sentences)
```

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

[45]:
```
text = "I will take the NLP class on Tuesday."
words = nltk.word_tokenize(text)
sentences = nltk.sent_tokenize(text)
```

[46]:
```
print(words)
print(sentences)
```

```
['I', 'will', 'take', 'the', 'NLP', 'class', 'on', 'Tuesday', '.']
['I will take the NLP class on Tuesday.']
```

3.0.1 Assign parts of speech tags to the sentencs

[47]:
```
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/jadi/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

[47]:
```
True
```

[48]:
```
tags = nltk.pos_tag(words)
print(tags)
```

```
[('I', 'PRP'), ('will', 'MD'), ('take', 'VB'), ('the', 'DT'), ('NLP', 'NNP'),
('class', 'NN'), ('on', 'IN'), ('Tuesday', 'NNP'), ('.', '.')]
```

3.0.2 Remove some of the stop words

[49]:
```
nltk.download('stopwords')
```

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

[49]:
```
True
```

[50]:
```
from nltk.corpus import stopwords

filtered_words = [word for word in words if word.lower() not in stopwords.
 ↪words('english')]

print(filtered_words)
```

```
['take', 'NLP', 'class', 'Tuesday', '.']
```