Chapter 10
Workshop#1 Basics of Natural Language Toolkit (Hour 1–2)

10.1 Introduction

Part 2 of this book will provide seven Python programming workshops on how each NLP core component operates and integrates with Python-based NLP tools including NLTK, spaCy, BERT, and Transformer Technology to construct a Q&A chatbot.

Workshop 1 will explore NLP basics including:

1. Concepts and installation procedures
2. Text processing function with examples using NLTK
3. Text analysis lexical dispersion plot in Python
4. Tokenization in text analysis
5. Statistical tools for text analysis

10.2 What Is Natural Language Toolkit (NLTK)?

NLTK (Natural Language Toolkit 2022) is one of the earliest Python-based NLP development tool invented by Prof. Steven Bird and Dr. Edward Loper in the Department of Computer and Information Science of University of Pennsylvania with their classical book Natural Language Processing with Python published by O'Reilly Media Inc. in 2009 (Bird et al. 2009). There are over 30 universities in USA and 25 countries using NLTK for NLP related courses until present. This book is considered as bible for anyone who wishes to learn and implement NLP applications using Python.

NLTK offers user-oriented interfaces with over 50 corpora and lexical resources such as WordNet, a large lexical database of English. Nouns, verbs, adjectives, and adverbs are grouped into sets of cognitive synonyms (synsets); each expresses a distinct concept which is an important lexical database in NLP developed by Princeton University since 1980.

Other lexical databases and corpora are Penn Treebank Corpus, Open Multilingual Wordnet, Problem Report Corpus, and Lin's Dependency Thesaurus.

NLTK contains statistical-based text processing libraries of five fundamental NLP enabling technologies and basic semantic reasoning tools including (Albrecht et al. 2020; Antic 2021; Arumugam and Shanmugamani 2018; Hardeniya et al. 2016; Kedia and Rasu 2020; Perkins 2014):

- Word tokenization
- Stemming
- POS tagging
- Text classification
- Semantic analysis

10.3 A Simple Text Tokenization Example Using NLTK

Let us look at NLTK text tokenization using Jupyter Notebook (Jupyter 2022; Wintjen and Vlahutin 2020) as below:

In[1]
```
# Import NLTK package
import nltk
```

In[2]
```
# Create a sample utterance 1 (utt1)
utt1 = "At every weekend, early in the morning. I drive my car to the car
center for car washing. Like clock-work."
```

In[3]
```
# Display utt1
utt1
```

Out[3]
```
'At every weekend, early in the morning. I drive my car to the car center
for car washing. Like clock-work.'
```

In[4]
```
# Create utterance tokens (utokens)
utokens = nltk.word_tokenize(utt1)
```

In[5]
```
# Display utokens
utokens
```

Out[5]
```
['At', 'every', 'weekend', ',', 'early', 'in', 'the', 'morning', '.', 'I', 'drive',
'my', 'car', 'to', 'the', 'car', 'center', 'for', 'car', 'washing', '.', 'Like', 'clock-
work', '.']
```

10.4 How to Install NLTK?

Step 1 Install Python 3.X

Step 2 Install NLTK

2.1 Start CMD or other command line tool

2.2 Type *pip install nltk*

Figure 10.1 shows a screenshot of NLTK installation process.

Step 3 Install NLTK Data

Once NLTK is installed into Python, download NLTK data.

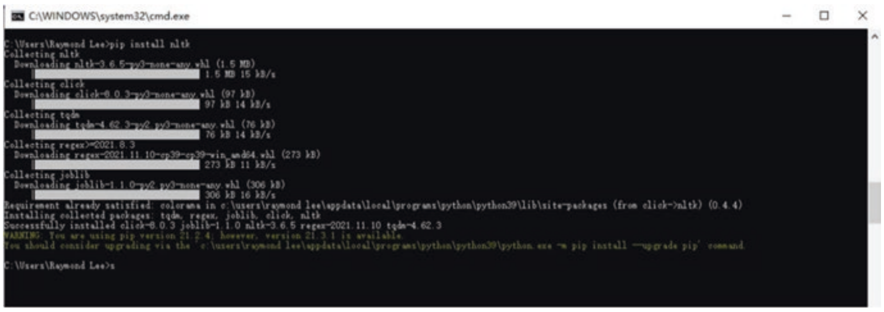
3.1 Run Python

3.2 Type the following to activate a NLTK downloader

import nltk

nltk.download()

Note: *nltk.downloader()* will invoke NLTK downloader automatically, a separate window-based downloading module for users to download four NLP databanks into their Python machines. They include (1) Collection libraries, (2) Corpora, (3) Modules, and (4) other NLP packages. Figures 10.2, 10.3, and 10.4 show screenshots of NTLK downloader for Collection, Corpora, and NTLK models installations.

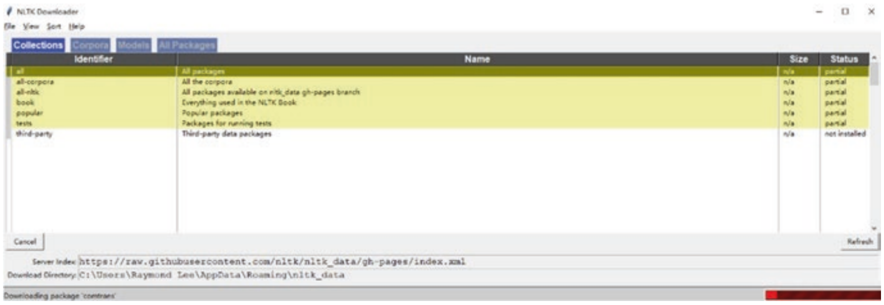


```
C:\WINDOWS\system32\cmd.exe

C:\Users\Raymond Lee>pip install nltk
Collecting nltk
  Downloading nltk-3.6.5-py3-none-any.whl (1.5 MB)
     |████████████████████████████████| 1.5 MB 15 kB/s
Collecting click
  Downloading click-8.0.3-py3-none-any.whl (97 kB)
     |████████████████████████████████| 97 kB 14 kB/s
Collecting tqdm
  Downloading tqdm-4.62.3-py2.py3-none-any.whl (76 kB)
     |████████████████████████████████| 76 kB 14 kB/s
Collecting regex>=2021.8.3
  Downloading regex-2021.11.10-cp39-cp39-win_amd64.whl (273 kB)
     |████████████████████████████████| 273 kB 11 kB/s
Collecting joblib
  Downloading joblib-1.1.0-py2.py3-none-any.whl (306 kB)
     |████████████████████████████████| 306 kB 16 kB/s
Requirement already satisfied: colorama in c:\users\raymond lee\appdata\local\programs\python\python39\lib\site-packages (from click->nltk) (0.4.4)
Installing collected packages: tqdm, regex, joblib, click, nltk
Successfully installed click-8.0.3 joblib-1.1.0 nltk-3.6.5 regex-2021.11.10 tqdm-4.62.3
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the 'c:\users\raymond lee\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip' command.

C:\Users\Raymond Lee>s
```

Fig. 10.1 Screenshot of NLTK installation process



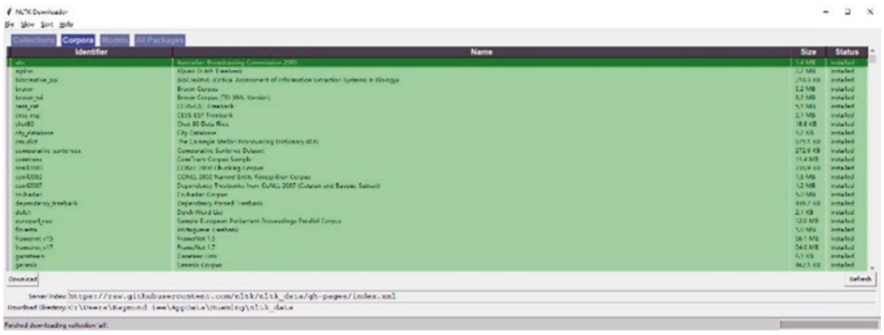
Identifier	Name	Size	Status
all	All packages	n/a	partial
all-corpora	All the corpora	n/a	partial
all-nltk	All packages available on nltk_data gh-pages branch	n/a	partial
book	Everything used in the NLTK Book	n/a	partial
popular	Popular packages	n/a	partial
tests	Packages for running tests	n/a	partial
third-party	Third-party data packages	n/a	not installed

Server Index: https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml

Download Directory: C:\Users\Raymond Lee\AppData\Roaming\nltk_data

Downloading package 'comtrans'

Fig. 10.2 Screenshot of NTLK downloader of Collection library

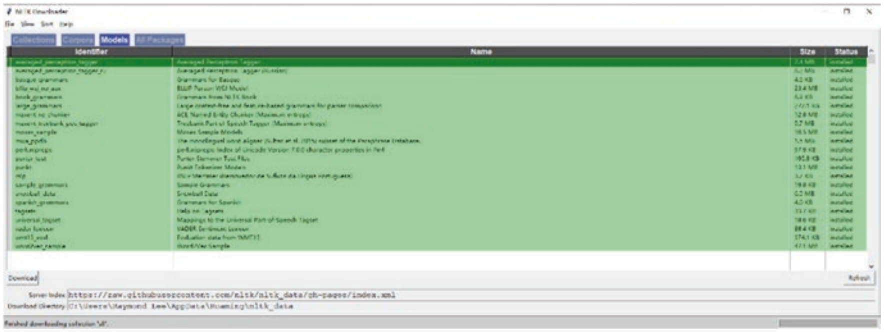


Identifier	Name	Size	Status
abc	Australian Broadcasting Commission 2000	1.4 MB	installed
alpino	Alpino Dutch Treebank	2.7 MB	installed
biocreative_ppi	BioCreAtIvE (Critical Assessment of Information Extraction Systems in Biology)	218.3 KB	installed
brown	Brown Corpus	3.2 MB	installed
brown_tei	Brown Corpus (TEI XML Version)	8.3 MB	installed
cess_cat	CESS-CAT Treebank	5.1 MB	installed
cess_esp	CESS-ESP Treebank	2.1 MB	installed
chat80	Chat-80 Data Files	18.8 KB	installed
city_database	City Database	1.7 KB	installed
cmudict	The Carnegie Mellon Pronouncing Dictionary (0.6)	875.1 KB	installed
comparative_sentences	Comparative Sentence Dataset	272.6 KB	installed
comtrans	ComTrans Corpus Sample	11.4 MB	installed
conll2000	CONLL 2000 Chunking Corpus	738.9 KB	installed
conll2002	CONLL 2002 Named Entity Recognition Corpus	1.8 MB	installed
conll2007	Dependency Treebanks from CoNLL 2007 (Catalan and Basque Subset)	1.2 MB	installed
crubadan	Crubadan Corpus	5.3 MB	installed
dependency_treebank	Dependency Parsed Treebank	445.7 KB	installed
dolch	Dolch Word List	2.1 KB	installed
europarl_raw	Sample European Parliament Proceedings Parallel Corpus	12.0 MB	installed
floresta	Portuguese Treebank	1.3 MB	installed
framenet_v15	FrameNet 1.5	56.1 MB	installed
framenet_v17	FrameNet 1.7	54.0 MB	installed
gazetteers	Gazeteer Lists	8.1 KB	installed
genesis	Genesis Corpus	862.1 KB	installed

Server Index: https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
Download Directory: C:\Users\Raymond Lee\AppData\Roaming\nltk_data

Finished downloading collection 'all'.

Fig. 10.3 Screenshot of NTLK downloader of Corpora library



Identifier	Name	Size	Status
averaged_perceptron_tagger	Averaged Perceptron Tagger	2.4 MB	installed
averaged_perceptron_tagger_ru	Averaged Perceptron Tagger (Russian)	8.2 MB	installed
basque_grammars	Grammars for Basque	4.2 KB	installed
bllip_wsj_no_aux	BLLIP Parser: WSJ Model	23.4 MB	installed
book_grammars	Grammars from NLTK Book	8.6 KB	installed
large_grammars	Large context-free and feature-based grammars for parser comparison	272.1 KB	installed
maxent_ne_chunker	ACE Named Entity Chunker (Maximum entropy)	12.8 MB	installed
maxent_treebank_pos_tagger	Treebank Part of Speech Tagger (Maximum entropy)	5.7 MB	installed
moses_sample	Moses Sample Models	10.5 MB	installed
mwa_ppdb	The monolingual word aligner (Sultan et al. 2015) subset of the Paraphrase Database.	1.5 MB	installed
perluniprops	perluniprops: Index of Unicode Version 7.0.0 character properties in Perl	97.9 KB	installed
porter_test	Porter Stemmer Test Files	195.8 KB	installed
punkt	Punkt Tokenizer Models	13.1 MB	installed
rslp	RSLP Stemmer (Removedor de Sufixos da Lingua Portuguesa)	3.7 KB	installed
sample_grammars	Sample Grammars	19.8 KB	installed
snowball_data	Snowball Data	6.2 MB	installed
spanish_grammars	Grammars for Spanish	4.0 KB	installed
tagsets	Help on Tagsets	33.7 KB	installed
universal_tagset	Mappings to the Universal Part-of-Speech Tagset	18.6 KB	installed
vader_lexicon	VADER Sentiment Lexicon	88.4 KB	installed
wmt15_eval	Evaluation data from WMT15	174.1 KB	installed
word2vec_sample	Word2Vec Sample	47.1 MB	installed

Server Index: https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
Download Directory: C:\Users\Raymond Lee\AppData\Roaming\nltk_data

Finished downloading collection 'all'.

Fig. 10.4 Screenshot of NTLK downloader of NLTK models

10.5 Why Using Python for NLP?

Python toolkit and packages overtook C, C++, Java especially in data science, AI, and NLP software development since 2000 (Albrecht et al. 2020; Kedia and Rasu 2020). There are several reasons to drive the changes because:

- it is a generic language without specific area unlike other language such as Java and JavaScript specifically designed for web applications and websites developments.
- it is easier to learn and user-friendly compared with C and C++ especially for non-computer science students and scientists.
- its lists and list-processing data types provide an ideal environment for NLP modelling and text analysis.

A Python program performs tokenization task to process text is shown below:

In[6]

```
# Define utterance 2 (utt2)
utt2 = "Hello world. How are you?"
```

In[7]

```
# Using split() method to split it into word tokens
utt2.split()
```

Out[7]

```
['Hello', 'world.', 'How', 'are', 'you?']
```

In[8]

```
# Check the no of word tokens
nwords = len(utt2.split())
print ("'Hello world. How are you?' contains ",nwords," words.")
```

Out[8]

```
'Hello world. How are you?' contains  5  words.
```

Python codes perform word number counts from literature *Alice's Adventures in Wonderland* by Lewis Carroll (1832–1898) as below:

In[9]

```
# Define method to count the number of word tokens in text file (cwords)
def cwords(literature):
    try:
        with open(literature, encoding='utf-8') as f_lit:
            c_lit = f_lit.read()
    except FileNotFoundError:
        err = "Sorry, the literature " + literature + " does not exist."
        print(err)
    else:
        w_lit = c_lit.split()
        nwords = len(w_lit)
         print("The literature " + literature + " contains " + str(nwords) +
" words.")

literature = 'alice.txt'
cwords(literature)
```

Out[9]

```
The literature alice.txt contains 29465 words
```

This workshop has extracted four famous literatures from Project Gutenberg (2022):
1. *Alice's Adventures in Wonderland* by Lewis Carroll (1832–1898) (alice.txt)
2. *Little Women* by Louisa May Alcott (1832–1888) (little_women.txt)
3. *Moby Dick* by Herman Melville (1819–1891) (moby_dick.txt)
4. *The Adventures of Sherlock Holmes* by Sir Arthur Conan Doyle (1859–1930) (Adventures_Holmes.txt)

In[10]
```
cwords('Adventures_Holmes.txt')
```
Out[10]
```
The literature Adventures_Holmes.txt contains 107411 words.
```

10.6 NLTK with Basic Text Processing in NLP

NLTK are Python tools and methods to learn and practice starting from basic text processing in NLP. They include:

- Text processing as lists of words
- Statistics on text processing
- Simple text analysis

NLTK provides 9 different types of text documents from classic literatures, Bible texts, famous public speeches, news, and articles with personal corpus for text processing. Let us start and load these text documents.

In[11]
```
# Let's load some sample books from the nltk databank
import nltk
from nltk.book import *
```
Out[11]
```
*** 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
```

In[12]
```
# Display the list of sample books
texts()
```

Out[12]
```
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
```

In[13]
```
# Check text1
text1
```

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

In[14]
```
# To know more about text1, check this
text1?
```

In[15]
```
# Import word_tokenize as wtoken
from nltk.tokenize import word_tokenize

# Open Adventures_Holmes.txt and performs tokenization
fholmes = open("Adventures_Holmes.txt","r",encoding="utf-8").read()
wtokens = word_tokenize(fholmes)
tholmes=nltk.text.Text(wtokens)
```

10.7 Simple Text Analysis with NLTK

Text analysis is to study a particular word or phrase occurred in a text document such as literature or public speeches. NLTK has a *"concordance()"* function different from ordinary search function. It does not only indicate occurrence but also reveal neighboring words and phrases. Let us try texts examples from *The Adventures of Sherlock Holmes* (Doyle 2019)

In[16]

```
# Check concordance of word "Sherlock "
tholmes.concordance("Sherlock")
```

Out[16]

```
Displaying 25 of 98 matches:
The Adventures of Sherlock Holmes by Arthur Conan Doyle Conte
es I . A SCANDAL IN BOHEMIA I . To Sherlock Holmes she is always _the_ woman .
ust such as I had pictured it from Sherlock Holmes ' succinct description , bu
ssing said : " Good-night , Mister Sherlock Holmes. " There were several peopl
lly got it ! " he cried , grasping Sherlock Holmes by either shoulder and look
stepped from the brougham . " Mr . Sherlock Holmes , I believe ? " said she .
ss for the Continent. " " What ! " Sherlock Holmes staggered back , white with
, the letter was superscribed to " Sherlock Holmes , Esq . To be left till cal
nd ran in this way : " MY DEAR MR. SHERLOCK HOLMES , —You really did it very w
 of interest to the celebrated Mr. Sherlock Holmes . Then I , rather imprudent
 possess ; and I remain , dear Mr. Sherlock Holmes , " Very truly yours , " IR
ia , and how the best plans of Mr. Sherlock Holmes were beaten by a woman ' s
 I had called upon my friend , Mr. Sherlock Holmes , one day in the autumn of
and discontent upon his features . Sherlock Holmes ' quick eye took in my occu
t as I have been telling you , Mr. Sherlock Holmes , " said Jabez Wilson , mop
e of this obliging youth ? " asked Sherlock Holmes . " His name is Vincent Spa
IS DISSOLVED . October 9 , 1890. " Sherlock Holmes and I surveyed this curt an
d client carried on his business . Sherlock Holmes stopped in front of it with
 own stupidity in my dealings with Sherlock Holmes . Here I had heard what he
" " I think you will find , " said Sherlock Holmes , " that you will play for
and I will follow in the second. " Sherlock Holmes was not very communicative
jump , and I ' ll swing for it ! " Sherlock Holmes had sprung out and seized t
IDENTITY " My dear fellow , " said Sherlock Holmes as we sat on either side of
ant-man behind a tiny pilot boat . Sherlock Holmes welcomed her with the easy
nsult me in such a hurry ? " asked Sherlock Holmes , with his finger-tips toge
```

The above example shows all *Sherlock* occurrences indicating that *Sherlock* is a special word linked with surname *Holmes* in text document.

Let's look at word usage of *extreme* from the same literature:

In[17]

```
# Check concordance of word "extreme"
tholmes.concordance("extreme")
```

Out[17]

```
Displaying 9 of 9 matches:
may trust with a matter of the most extreme importance . If not , I should much
ng red head , and the expression of extreme chagrin and discontent upon his fea
ternately asserted itself , and his extreme exactness and astuteness represente
e swing of his nature took him from extreme languor to devouring energy ; and ,
olice reports realism pushed to its extreme limits , and yet the result is , it
of an English provincial town . His extreme love of solitude in England suggest
ion , and that in his haste and the extreme darkness he missed his path and wal
for my coming at midnight , and his extreme anxiety lest I should tell anyone o
like one who has been driven to the extreme limits of his reason . Then , sudde
```

Concordance techniques are means to learn grammars, words or phrases called Use of English, also called Learn by Examples. In this example, we learnt how to use the word *extreme* in various situations and scenarios.

In[18]

```
tholmes.similar("extreme")
```

Out[18]

```
dense gathering
```

In[19]

```
# Check concordance of word "extreme" in text2
text2.concordance("extreme")
```

Out[19]

```
Displaying 4 of 4 matches:
n another day or two perhaps ; this extreme mildness can hardly last longer —
ng her that he was kept away by the extreme affection for herself , which he co
 of his brother , and lamenting the extreme GAUCHERIE which he really believed
y which had been leading her to the extreme of languid indolence and selfish re
```

In[20]

```
# Check similar word "extreme" in text2
text2.similar("extreme")
```

Out[20]

```
family centre good opinion life death loss house society children
attachment wishes interest goodness heart comfort cheerfulness
existence marriage son
```

In[21]

```
# Check concordance word "extreme" in text4
text4.concordance ("extreme")
```

Out[21]

```
Displaying 3 of 3 matches:
 vigilance no Administration by any extreme of wickedness or folly can very ser
ent , and communication between the extreme limits of the country made easier t
the politics of petty bickering and extreme partisanship they plainly deplore .
```

In[22]

```
# Check similar word "extreme" in text4
text4.similar("extreme")
```

Out[22]

```
one other just hope motives act people agency system right form loss
length knowledge science portion quarter narrowest requisite member
```

It showed that word usage of *extreme* varies by authors and text types e.g. it has different styles in *The Adventures of Sherlock Holmes* as compared with usage in *Sense and Sensibility* by Jane Austin (1775-1817) which is more vivid but has standard and fixed usage in *Inaugural Address Corpus*.

The *common_contexts()* method is to examine contexts shared by two or more words. *The Adventures of Sherlock Holmes* is used with common contexts of two words *extreme* and *huge*.

First, call common contexts() function from object *tholmes*.

In[23]

```
# Check common contexts on tholmes
tholmes.common_contexts(["extreme","huge"])
```

Out[23]

```
No common contexts were found
```

which means after analyzing *extreme* and *huge* in *The Adventures of Sherlock Holmes*, no common context meaning can be found.

Call *concordance()* function of these two words and check against the extracted patterns as shown below:

In[24]

```
# Check concordance word "extreme" in tholmes
tholmes.concordance("extreme")
```

Out[24]

```
Displaying 9 of 9 matches:
may trust with a matter of the most extreme importance . If not , I should much
ng red head , and the expression of extreme chagrin and discontent upon his fea
ternately asserted itself , and his extreme exactness and astuteness represente
e swing of his nature took him from extreme languor to devouring energy ; and ,
olice reports realism pushed to its extreme limits , and yet the result is , it
of an English provincial town . His extreme love of solitude in England suggest
ion , and that in his haste and the extreme darkness he missed his path and wal
for my coming at midnight , and his extreme anxiety lest I should tell anyone o
like one who has been driven to the extreme limits of his reason . Then , sudde
```

In[25]

```
# Check concordance word "huge" in tholmes
tholmes.concordance("huge")
```

Out[25]

```
Displaying 11 of 11 matches:
used and refreshed his memory with a huge pinch of snuff . " Pray continue you
 after opening a third door , into a huge vault or cellar , which was piled al
ed . All will come well . There is a huge error which it may take some little
 a small , office-like room , with a huge ledger upon the table , and a teleph
en suddenly dashed open , and that a huge man had framed himself in the apertu
 , and bent it into a curve with his huge brown hands . " See that you keep yo
r. Grimesby Roylott drive past , his huge form looming up beside the little fi
side and lay listless , watching the huge crest and monogram upon the envelope
 , " said I ruefully , pointing to a huge bundle in the corner . " I have had
th hanging jowl , black muzzle , and huge projecting bones . It walked slowly
r hurrying behind us . There was the huge famished brute , its black muzzle bu
```

Can you see how important it is in
1. NLP?
2. Use of English and technical writing?

Workshop 1.1 Simple Text Processing using NLTK
1. Try to use concordance(), similar(), and common_contexts() functions to look for two more frequently used words usage.
2. Compare their usages from four sources: *Moby Dick, Sense and Sensibility, Inaugural Address Corpus, and Wall Street Journal*.
3. Are there any pattern(s)?
4. What are their differences in Use of English?

10.8 Text Analysis Using Lexical Dispersion Plot

Text analysis was learnt to study word patterns and common contexts in previous workshop.

Dispersion Plot in Python NLTK is to identify occurrence frequencies of keywords from the whole document.

10.8.1 What Is a Lexical Dispersion Plot?

Dispersion is quantification of each point deviation from the mean value in basic statistics.

NLTK *Dispersion Plot* produces a plot showing words distribution throughout the text. *Lexical dispersion* is used to indicate homogeneity of words (word tokens) occurred in the corpus (text document) achieved by the dispersion_plot() in NLTK.

To start, let us use NLTK book object to call function *dispersion_plot()*.

Note: requires pylab installation prior this function.

The following example uses text1 to verify basic information about *dispersion_plot()*.

In[26]
```
text1.dispersion_plot?
```

10.8.2 Lexical Dispersion Plot Over Context Using Sense and Sensibility

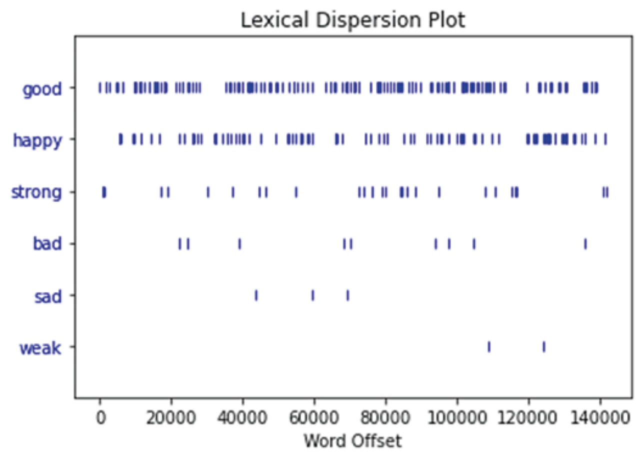
Are there any lexical patterns for positive words such as *good, happy*, and *strong* versus negative words such as *bad, sad,* or *weak* in literature?

Workshop 1.2 Lexical Dispersion Plot over Context using Sense and Sensibility
Use dispersion_plot to plot *Lexical Dispersion Plot* keywords: *good, happy, strong, bad, sad,* and *weak* from *Sense and Sensibility*.

1. Study any lexical pattern between positive and negative keywords.
2. Check these patterns against *Moby Dick* to see if this pattern occurs and explain.
3. Choose two other sentiment keywords to see if this pattern remains valid.

In[27]
```
text2.dispersion_plot(["good", "happy", "strong", "bad", "sad", "weak"])
```

Out[27]



10.8.3 *Lexical Dispersion Plot Over Time Using Inaugural Address Corpus*

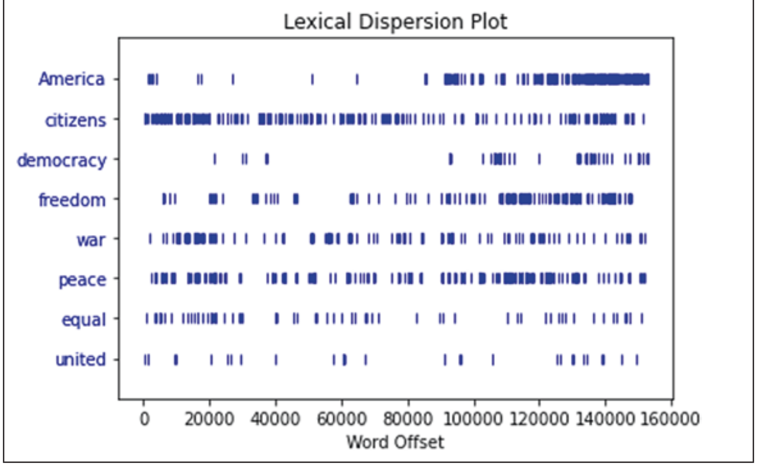
Lexical usage is to analyze word pattern changes in written English over time. The Inaugural Address Corpus addressed by US presidents of past 220 years is a text document in NLTK book library to study lexical dispersion plot patterns changes on keywords *war, peace, freedom,* and *united* for this workshop.

Workshop 1.3 Lexical Dispersion Plot over Time using Inaugural Address Corpus

1. Use dispersion_plot to invoke Lexical Dispersion Plot for *Inaugural Address Corpus*.
2. Study and explain lexical pattern changes for keywords *America*, *citizens*, *democracy*, *freedom*, *war*, *peace*, *equal*, *united*.
3. Choose any two meaningful keywords and check for lexical pattern changes.

In[28]
```
text4.dispersion_plot(["America" ,"citizens" ,"democracy",
"freedom", "war", "peace", "equal", "united"])
```

Out[28]



10.9 Tokenization in NLP with NLTK

10.9.1 What Is Tokenization in NLP?

A *token* can be words, part of a word, characters, numbers, punctuations, or symbols. It is a principal constituent and complex NLP task due to every language has own grammatical constructions to generate grammatic and syntactic rules.

Tokenization is an NLP process of dividing sentences/utterances from a text, document, or speeches into chunks called *tokens*. By using tokenization, a vocabulary from a document or corpus can be formed. Tokenization for sentence/utterances *Jane lent $100 to Peter early this morning* is shown in Fig. 10.5.

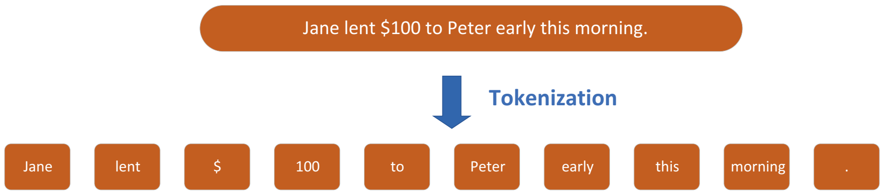


Fig. 10.5 Tokenization example of a sample utterance "*Jane lent $100 to Peter early this morning*"

NLTK provides flexibility to tokenize any string of text using *tokenize()* function as shown below:

In[29]
```
# Create utterance 3 (utt3) and performs tokenization
utt3 = 'Jane lent $100 to Peter early this morning.'
wtokens = nltk.word_tokenize(utt3)
wtokens
```
Out[29]
```
['Jane', 'lent', '$', '100', 'to', 'Peter', 'early', 'this', 'morning', '.']
```

10.9.2 Different Between Tokenize() vs Split()

Python provides *split()* function to split a sentence of text into words as recalled in Sect. 10.1 Let us see how it works with Tokenize() function.

In[30]
```
# Use split() to perform word tokenization
words = utt3.split()
words
```
Out[30]
```
['Jane', 'lent', '$100', 'to', 'Peter', 'early', 'this', 'morning.']
```

Why are they different?
How is it important in
1. NLP?
2. Meanings?

WORKSHOP

Workshop 1.4 Tokenization on The Adventures of Sherlock Holmes with NLTK
1. Read Adventures_Holmes.txt text file.
2. Save contents into a string object "holmes_doc".
3. Use split() to cut it into list object "holmes".
4. Count total number of words in the document.
5. Tokenize document using NLTK tokenize() function.
6. Count total number of tokens.
7. Compare the two figures.

(The file open part is provided to start with.)

In[31]
```
# Workshop 10.4 Solution

with open('Adventures_Holmes.txt', encoding='utf-8') as f_lit:
    dholmes = f_lit.read()
    # Count number of words in the literature
…
```

NLTK provides a simple way to count total number of tokens in a Text Document using len() in NLTK package.
Try len(tholmes) will notice:

```
In[32]   len(tholmes)
Out[32]  128366
```

10.9.3 Count Distinct Tokens

Text analysis is to study distinct words, or vocabulary occurred in a text document.
When text document is tokenized as token objects, Python can group them easily into a set of distinct object using Set() method.
Set() in Python is to extract distinct objects of any types from a list of objects with repeated instances.
Try the following using *The Adventures of Sherlock Holmes* will notice:

```
In[33]   tholmes?

In[34]   set(tholmes)
Out[34]  { 'fair', 'Absolutely', 'Castle', 'line', 'Street', 'nation', 'saturated', 'probing',
         'deposition', 'over-tender', 'obsolete', 'Sand.', 'confirmed', 'ponderous',
         'authoritative', 'Cosmopolitan', 'hopeless', 'white', 'eclipsed', 'furniture',
         'jury.', 'records', 'Coming', 'smear', 'ruined', 'hang', 'originator', 'ab-
         solved', 'hole', 'notorious', 'perfectly', 'worker', ... }

In[35]   len(set(tholmes))
Out[35]  10048
```

This example showed that *The Adventures of Sherlock Holmes* contains 128,366 tokens, i.e. words and punctuations, and 10,048 distinct tokens, or types. Try other literatures and see vocabulary can be learnt from these great literatures.

The following example shows how to sort distinct tokens using sorted() function.

```
In[36]   sorted(set(tholmes))

Out[36]  { '!', '$', '%', '&', "'", "''", "'AS-IS", "'s", '(', ')', '*', ',', '-', '--', '-the-
         wisp', '.', '1', '1,100', '1.A', '1.B', '1.C', '1.D', '1.E', '1.E.1', '1.E.2',
          '1.E.3', '1.E.4', '1.E.5', '1.E.6', '1.E.7', '1.E.8', ... }
```

Since books are tokenized in NLTK as a list book object, contents can be accessed by using list indexing method as below:

```
In[37]   # Access the First 20 tokens
         tholmes[1:20]

Out[37]  ['Adventures', 'of', 'Sherlock', 'Holmes', 'by', 'Arthur', 'Conan', 'Doyle',
          'Contents', 'I', '.', 'A', 'Scandal', 'in', 'Bohemia', 'II', '.', 'The',
          'Red-Headed']

In[38]   # Access the MIDDLE content
         tholmes[100:150]

Out[38]  ['IN', 'BOHEMIA', 'I', '.', 'To', 'Sherlock', 'Holmes', 'she', 'is', 'always',
          '_the_', 'woman', '.', 'I', 'have', 'seldom', 'heard', 'him', 'mention', 'her']

In[39]   # Aceess from the END
         tholmes[-20:]

Out[39]  ['help', 'produce', 'our', 'new', 'eBooks', ',', 'and', 'how', 'to', 'subscribe',
          'to', 'our', 'email', 'newsletter', 'to', 'hear', 'about', 'new', 'eBooks', '.']
```

10.9.4 *Lexical Diversity*

10.9.4.1 Token Usage Frequency (Lexical Diversity)

Token usage frequency, also called *Lexical Diversity* is to divide the total number of tokens by total number of token types as shown:

```
In[40]   len(text1)/len(set(text1))

Out[40]  13.502044830977896
```

In[41]
```
len(text2)/len(set(text2))
```
Out[41]
```
20.719449729255086
```

In[42]
```
len(text3)/len(set(text3))
```
Out[42]
```
16.050197203298673
```

In[43]
```
len(text4)/len(set(text4))
```
Out[43]
```
15.251970074812968
```

Python codes above analyze token usage frequency of four literatures: *Moby Dick, Sense and Sensibility, Book of Genesis, and Inaugural Address Corpus.* It has usage frequency range from 13.5 to 20.7. What are the implications?

10.9.4.2 Word Usage Frequency

There are many commonly used words in English. The following example shows the pattern of word usage frequency for *the* from above literatures.

In[44]
```
text1.count('the')
```
Out[44]
```
13721
```

In[45]
```
text1.count('the')/len(text1)*100
```
Out[45]
```
5.260736372733581
```

In[46]
```
text2.count('the')/len(text2)*100
```
Out[46]
```
2.7271571452788606
```

In[47]
```
text3.count('the')/len(text3)*100
```
Out[47]
```
5.386024483960325
```

In[48]
```
text4.count('the')/len(text4)*100
```
Out[48]
```
6.2491416014283745
```

1. Are there any patterns found from these literatures?
2. Use other words *of*, *a*, *I* to study if there exists other pattern(s).

10.10 Basic Statistical Tools in NLTK

10.10.1 Frequency Distribution: FreqDist()

Text analysis is a NTLK tool that can tokenize a string or a book of text document.

Frequency Distribution—*FreqDist()* is an initial built-in method in NLTK to analyze frequency distribution of every token type in a text document.

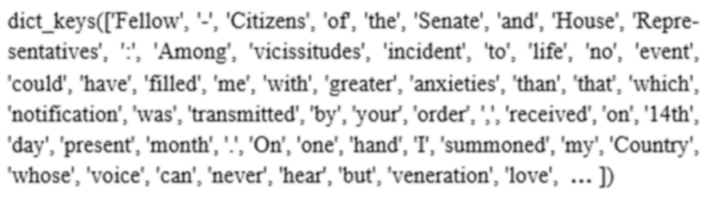
Inaugural Address Corpus is used as an example to show how it works.

```
In[49]    text4
Out[49]   <Text: Inaugural Address Corpus>
In[50]    FreqDist?
In[51]    fd4 = FreqDist(text4)
In[52]    fd4
Out[52]   FreqDist({'the': 9555, ',': 7275, 'of': 7169, 'and': 5226, '.': 5011, 'to': 4477, 'in':
          2604, 'a': 2229, 'our': 2062, 'that': 1769, ...})
```

10.10.1.1 FreqDist() as Dictionary Object

It is noted that *FreqDist()* will return *key-value* pairs from *Dictionary* object to reflect the *Key* that store *Token Type* name and the *Value* which are corresponding frequency of occurrence in a text. Since *FreqDist()* returns a *Dictionary* object, *keys()* can be used to return the list of all *Token Types* as shown below.

```
In[53]    token4 = fd4.keys()
          token4
Out[53]
```



```
dict_keys(['Fellow', '-', 'Citizens', 'of', 'the', 'Senate', 'and', 'House', 'Repre-
sentatives', ':', 'Among', 'vicissitudes', 'incident', 'to', 'life', 'no', 'event',
'could', 'have', 'filled', 'me', 'with', 'greater', 'anxieties', 'than', 'that', 'which',
'notification', 'was', 'transmitted', 'by', 'your', 'order', ',', 'received', 'on', '14th',
'day', 'present', 'month', '.', 'On', 'one', 'hand', 'I', 'summoned', 'my', 'Country',
'whose', 'voice', 'can', 'never', 'hear', 'but', 'veneration', 'love', ... ])
```

10.10.1.2 Access FreqDist of Any Token Type

Use list item access method to obtain frequency distribution of any token types. FD value of token type for *the* is shown below.

```
In[54]   fd4['the']
Out[54]  9555
```

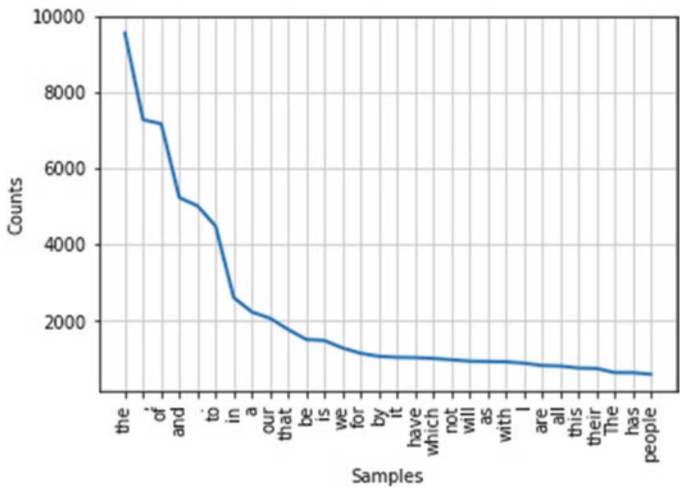
1. What are five common word types (token types without punctuations) in any text document?
2. Use FreqDist() to verify.

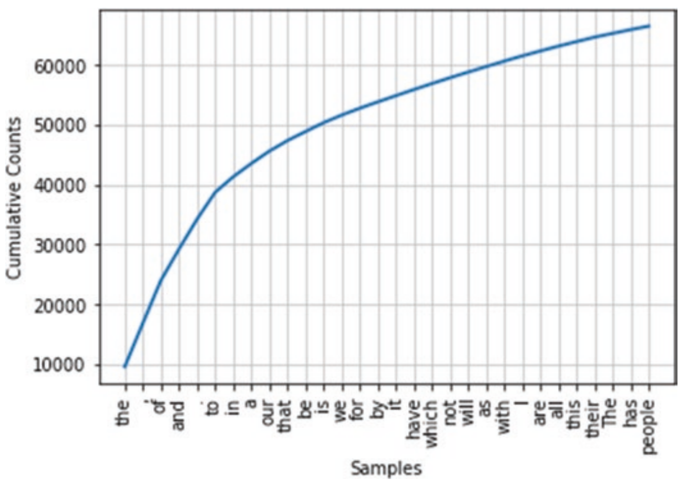
10.10.1.3 Frequency Distribution Plot from NLTK

NLTK is a useful tool to study the top frequency distribution token types for any document using plot() function with FreqDist() method. FreqDist.plot() can also plot the top XX frequently used token types in a text document.

1. Use fd3 to study *FreqDist.plot()* documentation using *fd3.plot()*.
2. Plot top 30 frequently used token types from the *Book of Genesis* (Non-Cumulative mode).
3. Do the same plot with *Cumulative* mode.

```
In[55]   fd4.plot?
In[56]   fd4.plot (30,cumulative=False)
Out[56]
```



In[57]
```
fd4.plot (30,cumulative=True)
```
Out[57]


Workshop 1.5 Frequency Distribution Analysis on Classics Literatures

1. What are top 5 frequently used word types in the *Book of Genesis* (ignore punctuations)?
2. Will it be the same with other great literatures?
3. Verify against *(1) Moby Dick, (2) Sense and Sensibility, and (3) Inaugural Address Corpus* to see if they have the same patterns. Why or why not?
4. Why the study of common word types is also important in cryptography?

10.10.2 *Rare Words: Hapax*

Hapaxes are words that occur only once in a body of work whether it is a publication or an entire language.

Ancient texts are full of hapaxes. For instance, in Shakespeare's *Love's Labour's Lost* contains hapax *honorificabilitudinitatibus* which means able to achieve honors.

NLTK provides method hapaxes() under FreqDist object to list out all word types that occurred once in text document.

Try FreqDist() with *The Adventures of Sherlock Holmes* and see how useful it is.

In[58]
```
tholmes
```
Out[58]
```
<Text: The Adventures of Sherlock Holmes by Arthur Conan...>
```
In[59]
```
fd = FreqDist(tholmes)
```

In[60]
```
hap = fd.hapaxes()
hap[1:50]
```

Out[60]
```
['Adventures', 'Conan', 'Doyle', 'Contents', 'Red-Headed', 'Case',
'Identity', 'Mystery', 'Orange', 'Pips', 'Twisted', 'Lip', 'Blue', 'Carbuncle',
'Speckled', 'Band', 'Engineer', 'Thumb', 'Noble', 'Bachelor', 'SCANDAL',
'BOHEMIA', 'eclipses', 'predominates', 'sex', 'emotions', 'abhorrent',
'balanced', 'softer', 'passions', 'gibe', 'observer—excellent', 'intrusions',
'finely', 'temperament', 'distracting', 'mental', 'Grit', 'sensitive',
'instrument', 'high-power', 'lenses', 'disturbing', 'dubious', 'home-centred',
'establishment', 'absorb', 'loathed', 'alternating' ]
```

WORKSHOP

Workshop 1.6 Learn Vocabulary using Hapaxes
Hapaxes are useful for us to learn vocabulary contain more than 12 characters. The following example uses hapaxes() with Python in-line function to implement [w for w in hap1 if len(w) > 12]:

1. Create the Python script and extract vocabulary contain more than 12 characters from *Moby Dick*.
2. Select five meaningful vocabularies with their meanings.
3. Check with *The Adventures of Sherlock Holmes* to learn another five vocabularies.

(Python script to generate vocabulary with over 12 characters are given.)

In[61]
```
# Workshop 10.6 Solutions

voc12 = [w for w in hap if len(w) > 12]
voc12
```

Out[61]
```
['observer—excellent', 'establishment', 'well-remembered', 'boot-slitting',
'Peculiar—that', 'Eglonitz—here', 'German-speaking', 'glass-factories',
'authoritative', 'double-breasted', 'Cassel-Felstein', 'staff-commander',
'Contralto—hum', 'indiscretion.', 'reproachfully', 'Saxe-Meningen',
'drunken-looking', 'side-whiskered', 'half-and-half, ... ]
```

10.10.3 Collocations

10.10.3.1 What Are Collocations?

A *collocation* is a work grouping for a set of words usually appeared together to convey semantic meanings. The word *collocation* is originated from Latin word meaning place together and was first introduced by Prof. John R Firth (1890–1960) with his famous quote "You shall know a word by the company it keeps."

There are many cases in English where strong collocations are word pairings always appear together such as *make* and *do,* e.g. You make a cup of coffee, but you do your work.

Collocations are frequently used in business settings when nouns are combined with verbs or adjectives, e.g. set up an appointment, conduct a meeting, set the price etc.

10.10.3.2 Collocations in NLTK

NLTK also provides a build-in method to handle collocations using NLTK method—*collocations().*

The following example is to generate collocations lists from *Moby Dick, Sense and Sensibility*, *Book of Genesis,* and *Inaugural Address Corpus*.

Let us look at some extracted collocation terms:

In[62]
```
text1.collocations()
```

Out[62]
```
Sperm Whale; Moby Dick; White Whale; old man; Captain Ahab; sperm
whale; Right Whale; Captain Peleg; New Bedford; Cape Horn; cried Ahab;
years ago; lower jaw; never mind; Father Mapple; cried Stubb; chief
mate; white whale; ivory leg; one hand
```

In[63]
```
text2.collocations()
```

Out[63]
```
Colonel Brandon; Sir John; Lady Middleton; Miss Dashwood; every thing;
thousand pounds; dare say; Miss Steeles; said Elinor; Miss Steele;
every body; John Dashwood; great deal; Harley Street; Berkeley Street;
Miss Dashwoods; young man; Combe Magna; every day; next morning
```

In[64]
```
text3.collocations()
```

Out[64]
```
said unto; pray thee; thou shalt; thou hast; thy seed; years old;
spake unto; thou art; LORD God; every living; God hath; begat sons;
seven years; shalt thou; little ones; living creature; creeping thing;
savoury meat; thirty years; every beast
```

In[65]
```
text4.collocations()
```

Out[65]
```
United States; fellow citizens; years ago; four years; Federal
Government; General Government; American people; Vice President; God
bless; Chief Justice; one another; fellow Americans; Old World;
Almighty God; Fellow citizens; Chief Magistrate; every citizen; Indian
tribes; public debt; foreign nations
```

References

Albrecht, J., Ramachandran, S. and Winkler, C. (2020) Blueprints for Text Analytics Using Python: Machine Learning-Based Solutions for Common Real World (NLP) Applications. O'Reilly Media.

Antic, Z. (2021) Python Natural Language Processing Cookbook: Over 50 recipes to understand, analyze, and generate text for implementing language processing tasks. Packt Publishing.

Arumugam, R. and Shanmugamani, R. (2018) Hands-On Natural Language Processing with Python: A practical guide to applying deep learning architectures to your NLP applications. Packt Publishing.

Bird, S., Klein, E., and Loper, E. (2009). Natural language processing with python. O'Reilly.

Doyle, A. C. (2019) The Adventures of Sherlock Holmes (AmazonClassics Edition). AmazonClassics.

Gutenberg (2022) Project Gutenberg official site. https://www.gutenberg.org/ Accessed 16 June 2022.

Hardeniya, N., Perkins, J. and Chopra, D. (2016) Natural Language Processing: Python and NLTK. Packt Publishing.

Jupyter (2022) Jupyter official site. https://jupyter.org/. Accessed 16 June 2022.

Kedia, A. and Rasu, M. (2020) Hands-On Python Natural Language Processing: Explore tools and techniques to analyze and process text with a view to building real-world NLP applications. Packt Publishing.

NLTK (2022) NLTK official site. https://www.nltk.org/. Accessed 16 June 2022.

Perkins, J. (2014). Python 3 text processing with NLTK 3 cookbook. Packt Publishing Ltd.

Wintjen, M. and Vlahutin, A. (2020) Practical Data Analysis Using Jupyter Notebook: Learn how to speak the language of data by extracting useful and actionable insights using Python. Packt Publishing.

WordNet (2022) WordNet official site. https://wordnet.princeton.edu/. Accessed 16 June 2022.