<u>Tutorial 01</u>

# Import the Natural Language Toolkit (NLTK) library

import nltk

# Downloads a collection of example texts (called corpora) used for learning. These are built-in sample books from NLTK

nltk.download('book')

# Import all the preloaded texts like text1, text2, ..., text9

from nltk.book import *

# Print which book is loaded in text1

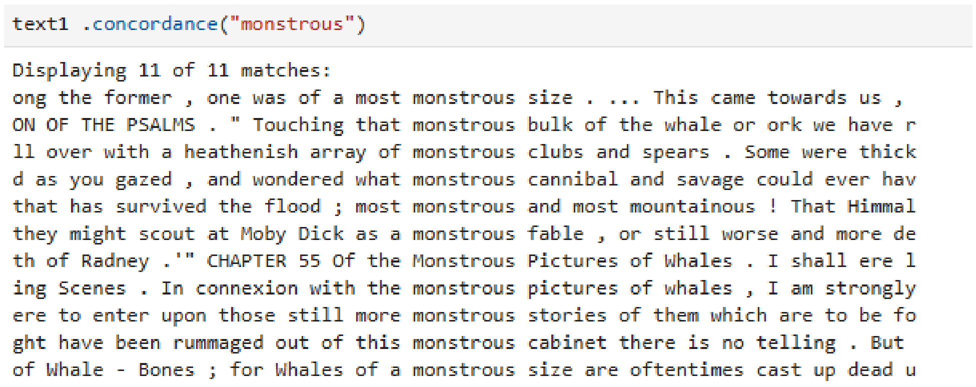
print(text1) # Output: <Text: Moby Dick by Herman Melville 1851>

NLTK Text object –

behaves like a list of words.

# Show all occurrences of the word "monstrous" in text1 along with surrounding words

text1.concordance("monstrous")



```
text1 .concordance("monstrous")
```

```
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
```

# Count and print how many times the word "whale" appears in the book

print("Frequency of 'whale':", text1.count("whale"))

# Calculate and print the vocabulary size (number of unique words) in text1

print("Vocabulary size:", len(set(text1)))

unique

chatgpt example:

```
# Load necessary modules from NLTK
import nltk
from nltk.text import Text
from nltk.tokenize import word_tokenize
nltk.download('punkt')  # tokenizer models

# ----------- Custom Text (replace this with your own) -------------
custom_text = """
Call me Ishmael. Some years ago—never mind how long precisely—having little or no money in my purse,
and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world.
It is a way I have of driving off the spleen and regulating the circulation.
"""

# ----------- Tokenization -------------
# Convert the paragraph into a list of word tokens
tokens = word_tokenize(custom_text)

# Wrap tokens in NLTK Text object (adds concordance and other methods)
text_custom = Text(tokens)

# ----------- Analysis -----------------

# Show the object (optional)
print(text_custom)  # Output: <Text: Call me Ishmael . Some years ago — never... >
```

```
# Concordance: Show each appearance of the word "sail" and its surrounding context

text_custom.concordance("sail")


# Frequency of a word (remove extra space in "whale" if copying from your code)

print("Frequency of 'whale':", text_custom.count("whale"))


# Vocabulary size (number of unique words)

print("Vocabulary size:", len(set(text_custom)))
```

Tutorial 02

Text Analysis Techniques in NLTK

A. Concordance Search

```
text1.concordance("monstrous")

text2.concordance("affection")
```

Theory:

- **concordance(word)** finds and displays all **occurrences** of a given word in a text.
- It shows the **context** in which the word appears (a few words before and after).
- Helps you understand the **meaning and usage** of the word in different situations.

B. Finding Similar Words

```
text1.similar("monstrous")

text2.similar("affection")
```

Theory:

What does text1.similar("monstrous") do?

- It finds words that appear in contexts similar to the word "monstrous" in the text.
- For example, if "monstrous" appears in sentences like:
  "the monstrous whale" or *"the monstrous size"*,

then similar("monstrous") finds other words that show up in similar positions in sentences — like "huge", "gigantic", "fearful", etc.

C. Vocabulary Analysis

# Get sorted unique tokens

sorted_unique_tokens = sorted (set (text3))

print (sorted_unique_tokens [:30]) # First 30 tokens

# Count total and unique tokens

total_tokens = len (text3)

unique_tokens = len (set (text3))

# Calculate lexical diversity

lexical_diversity = ~~total_tokens / unique_tokens~~ lexical_diversity = unique_tokens / total_tokens

print (" Lexical Diversity : " , lexical_diversity)

What is Lexical Diversity?

- It is a measure of **how varied the vocabulary** is in a text.
- In other words, it tells you **how many different unique words** are used compared to the total number of words.

Why is it important?

- A text with **high lexical diversity** uses a **wide range of different words** — it's more rich and varied.
- A text with **low lexical diversity** repeats the same words many times — it's more repetitive or simple.

D. Vocabulary Analysis

# Count occurrences of a word

whale_count = text3.count("smote")

print("Occurrences of 'smote':", whale_count)