# Natural Language Processing using Python Programming

## Notebook 03.1: Exploring NLTK Corpora


---

**Part of the comprehensive learning series:** Natural Language Processing using Python Programming

**Learning Objectives:**

- Explore and utilize NLTK's built-in text corpora for linguistic analysis
- Master frequency distribution analysis using FreqDist for vocabulary insights
- Implement concordance and collocation techniques for contextual word study
- Analyze different text genres and categories using the Brown corpus
- Build foundation skills for working with large text collections

---

- A **Corpus** (plural: **Corpora**) is a large, structured set of text, often annotated, used for linguistic analysis.
- NLTK provides access to many built-in corpora, which are perfect for quickly exploring language patterns and demonstrating foundational NLP techniques like tokenization and frequency analysis.

## 1. Setting up: Importing Libraries and Corpora

- We need to import the necessary NLTK corpus modules and ensure the required data is downloaded.

In [1]:
```
# Import necessary libraries
# gutenberg, brown, reuters, movie_reviews are popular NLTK corpora
# We will use these corpora to demonstrate tokenization and other NLP tasks
import nltk
from nltk.corpus import gutenberg, brown, reuters, movie_reviews
from nltk.tokenize import word_tokenize

# Download necessary resources (If not done in 1.2)
nltk.download(['gutenberg', 'brown', 'reuters', 'movie_reviews', 'punkt'], quiet=1

print("NLTK Corpora loaded and ready.")
```

```
NLTK Corpora loaded and ready.
```

---

## 2. Accessing and Exploring Corpora

- NLTK corpora are organized by files, words, sentences, and categories, allowing easy retrieval of raw data.

## 2.1 The Gutenberg Corpus

- The Gutenberg corpus is a selection of 18 famous texts from Project Gutenberg (e.g., Jane Austen, Shakespeare).

In [3]:
```
# gutenberg corpus exploration with fileids, words, and sents
gutenberg_files = gutenberg.fileids()
print(f"Gutenberg files: {gutenberg_files}\n")
```

```
Gutenberg files: ['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt',
'bible-kjv.txt', 'blake-poems.txt', 'bryant-stories.txt', 'burgess-busterbrown.tx
t', 'carroll-alice.txt', 'chesterton-ball.txt', 'chesterton-brown.txt', 'chesterton
-thursday.txt', 'edgeworth-parents.txt', 'melville-moby_dick.txt', 'milton-paradis
e.txt', 'shakespeare-caesar.txt', 'shakespeare-hamlet.txt', 'shakespeare-macbeth.tx
t', 'whitman-leaves.txt']
```

In [5]:
```
# Analyzing Jane Austen's 'Emma'
emma_words = gutenberg.words('austen-emma.txt')
print(f"Total words in Emma: {len(emma_words)}")
print(f"First 10 words: {emma_words[:10]}")
```

```
Total words in Emma: 192427
First 10 words: ['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']', 'VOLUME', 'I',
'CHAPTER']
```

In [9]:
```
# Analyzing Jane Austen's 'Emma'
emma_sents = gutenberg.sents('austen-emma.txt')

print(f"Total sentences in Emma: {len(emma_sents)}")
print(f"First 3 sentences: {emma_sents[:3]}")
```

```
Total sentences in Emma: 7752
First 3 sentences: [['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']'], ['VOLUME',
'I'], ['CHAPTER', 'I']]
```

## 2.2 The Brown Corpus

- The Brown Corpus was the first electronically prepared corpus, categorized by genre (e.g., news, religion, science fiction).

- This allows for comparative linguistic analysis.

In [16]:
```
# Exploring the Brown Corpus with categories and words
brown_categories = brown.categories()
print(f"Brown Categories: {brown_categories}\n")
```

```
Brown Categories: ['adventure', 'belles_lettres', 'editorial', 'fiction', 'governme
nt', 'hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'review
s', 'romance', 'science_fiction']
```

**Python code to iterate through all categories and print word counts and first 10 words**

```
for category in brown_categories:
    category_words = brown.words(categories=category)
    print(f"Category: {category}, Number of words:
{len(category_words)}")
    print(f"First 10 words in '{category}':
{category_words[:10]}\n")
```

In [11]:
```
# Accessing words specifically from the 'news' category
news_words = brown.words(categories='news')

print(f"Words in 'news' category: {len(news_words)}")
print(f"First 10 words from 'news': {news_words[:10]}")
```

```
Words in 'news' category: 100554
First 10 words from 'news': ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'F
riday', 'an', 'investigation', 'of']
```

---

## 3. Frequency Distribution: The `FreqDist` Object

- The **Frequency Distribution ( `FreqDist` )** is a powerful NLTK tool that counts the occurrences of all items (words, letters, etc.) in a text.
- It provides the statistical backbone for many text analyses.

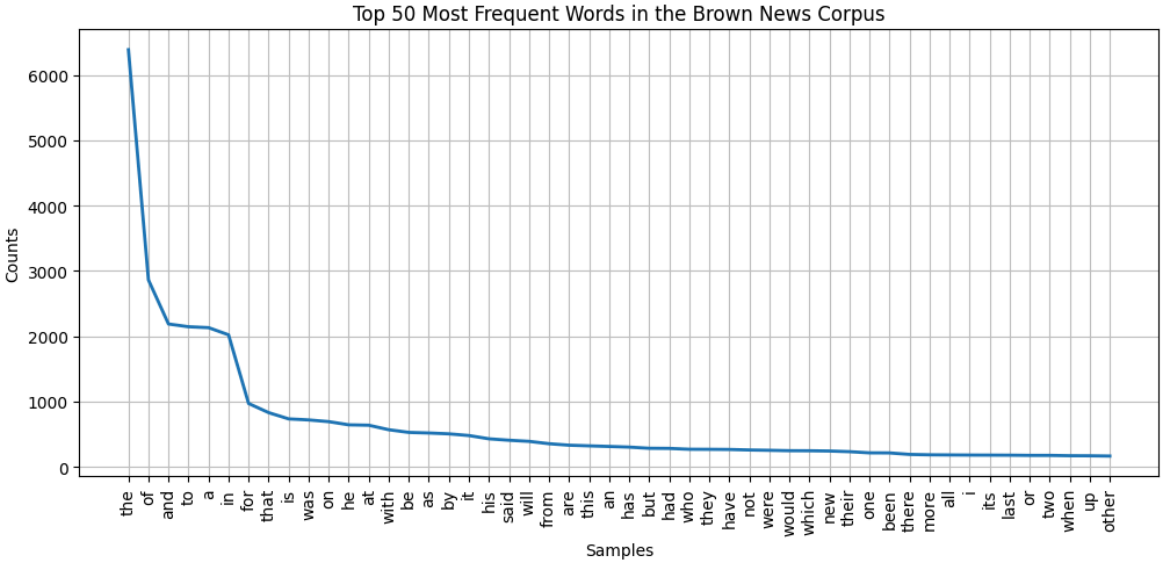
In [17]:
```
# Import FreqDist for frequency distribution analysis
# We will create a frequency distribution of words in the 'news' category of the B
# probability module provides the FreqDist class
from nltk.probability import FreqDist

# 1. Clean and normalize the words from the 'news' category (lower-casing)
news_words_lower = [w.lower() for w in news_words if w.isalpha()]

# 2. Create the Frequency Distribution object
fdist = FreqDist(news_words_lower)

print(f"Total number of unique words: {len(fdist)}\n")

# Display the 10 most common words
print("10 Most Common Words:")
for word, frequency in fdist.most_common(10):
    print(f" - {word:<10}: {frequency}")
```

```
Total number of unique words: 11151

10 Most Common Words:
 - the       : 6386
 - of        : 2861
 - and       : 2186
 - to        : 2144
 - a         : 2130
 - in        : 2020
 - for       : 969
 - that      : 829
 - is        : 733
 - was       : 717
```

### Frequency Distribution Visualization

- We can easily visualize the frequency distribution to understand the distribution of vocabulary.

In [ ]:
```
# Visualizing the frequency distribution
# Import matplotlib for plotting
import matplotlib.pyplot as plt

# Set up the plot size and title
plt.figure(figsize=(12, 5))
plt.title('Top 50 Most Frequent Words in the Brown News Corpus')
# Plot the 50 most common words
fdist.plot(50, cumulative=False)
plt.show()
# cumulative=False for non-cumulative plot means we see the raw counts
```




> **Observation:** The plot confirms that the most frequent words are generally **stopwords** ('the', 'of', 'and'), highlighting why stopword removal (Chapter 2.1) is crucial when preparing data for machine learning.

---

## 4. Collocations and Concordance

- These tools help us study the *context* and *relationships* between words.

### 4.1 Concordance

- **Concordance** shows every occurrence of a given word, together with some surrounding context.
- This is useful for manual text exploration.

In [19]:
```
# Import Text class from nltk.text for concordance analysis
from nltk.text import Text
```

```
# Create an NLTK Text object from a book
emma_text = Text(gutenberg.words('austen-emma.txt'))

print("Concordance for the word 'gentleman':")
emma_text.concordance("gentleman")
```

```
Concordance for the word 'gentleman':
Displaying 25 of 35 matches:
re can be no doubt of your being a gentleman ' s daughter , and you must suppor
Knightley . But he is not the only gentleman you have been lately used to . Wha
tion was most suitable , quite the gentleman himself , and without low connexio
"-- unclosing a pretty sketch of a gentleman in small size , whole - length --"
tion it would not have disgraced a gentleman ; the language , though plain , wa
ied to a respectable , intelligent gentleman - farmer !" " As to the circumstan
ly be a doubt that her father is a gentleman -- and a gentleman of fortune .--
her father is a gentleman -- and a gentleman of fortune .-- Her allowance is ve
ement or comfort .-- That she is a gentleman ' s daughter , is indubitable to m
 gentlemen are ; and nothing but a gentleman in education and manner has any ch
 , " Oh ! dear , yes ," before the gentleman joined them . The wants and suffer
. Mr . John Knightley was a tall , gentleman - like , and very clever man ; ris
modern days indeed have rendered a gentleman ' s carriage perfectly complete .
countenance was necessary for each gentleman as they walked into Mrs . Weston '
 of her other complaint before the gentleman ' s return . She went to Mrs . God
was _very_ _well_ _married_ , to a gentleman in a _great_ _way_ , near Bristol
u should do ," said she ; " like a gentleman .-- I am quite glad to see you ."
 have discerned me to be more of a gentleman than usual .-- You might not have
ligence passed between her and the gentleman on first glancing towards Miss Fai
is friend . Knightley is quite the gentleman . I like him very much . Decidedly
uch . Decidedly , I think , a very gentleman - like man ." Happily , it was now
tley !-- and discover that he is a gentleman ! A little upstart , vulgar being
 discover that Mr . Knightley is a gentleman ! I doubt whether he will return t
a smile at her . " I never saw any gentleman ' s handwriting "-- Emma began , l
k Churchill writes one of the best gentleman ' s hands I ever saw ." " I do not
```

### 4.2 Collocations

- **Collocations** are sequences of words that frequently occur together (e.g., 'red tape', 'New York').
- NLTK's function attempts to identify statistically significant co-occurring words.

In [22]:
```
print("Top 10 Collocations in 'Emma':\n")

emma_text.collocations(10)
```

```
Top 10 Collocations in 'Emma':

Frank Churchill; Miss Woodhouse; Miss Bates; Jane Fairfax; Miss
Fairfax; every thing; young man; every body; great deal; dare say
```

## 5. Summary and Next Steps

- NLTK's corpora provide an excellent sandbox for learning to access, count, and analyze text.
- We utilized `FreqDist` for vocabulary analysis and `concordance` / `collocations` for context.

- In the next notebook (3.2), we move from NLTK's static collections to the data science standard: **Loading real-world, large datasets using Pandas** and performing initial Exploratory Data Analysis (EDA).

### Key Takeaways

- **Corpora Exploration:** We successfully explored NLTK's built-in corpora including Gutenberg, Brown, Reuters, and Movie Reviews, learning how to access structured text collections.
- **Frequency Analysis:** We mastered FreqDist for statistical vocabulary analysis, discovering the dominance of stopwords and the importance of text preprocessing.
- **Contextual Analysis:** We implemented concordance and collocation techniques to understand word relationships and contextual usage patterns.

---

## *Next Notebook Preview*

- Now that we've mastered working with NLTK's built-in corpora, we're ready to work with **real-world datasets**.
- The next notebook will focus on **loading and analyzing large text datasets using Pandas**, performing exploratory data analysis (EDA) on modern text collections.

---

### About This Project

This notebook is part of the **Natural Language Processing using Python Programming for Beginners** repository - a comprehensive, beginner-friendly guide for mastering NLP using Python, NLTK, and SpaCy.

**Repository:** `NLP`

### Author

**Prakash Ukhalkar**

GitHub prakash-ukhalkar

---