Text analysis and natural language processing with NLTK

In [2]:
```
import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

In [4]:
```
# pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\nms31\anaconda3\lib\site-packages (3.8.1)
Requirement already satisfied: click in c:\users\nms31\anaconda3\lib\site-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in c:\users\nms31\anaconda3\lib\site-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in c:\users\nms31\anaconda3\lib\site-packages (from nltk) (2023.10.3)
Requirement already satisfied: tqdm in c:\users\nms31\anaconda3\lib\site-packages (from nltk) (4.66.4)
Requirement already satisfied: colorama in c:\users\nms31\anaconda3\lib\site-packages (from click->nltk) (0.4.6)
Note: you may need to restart the kernel to use updated packages.
```

In [37]:
```
import nltk
nltk.download('punkt')  # For tokenization
nltk.download('stopwords')  # For stopword removal
nltk.download('wordnet')  # For lemmatization
nltk.download('averaged_perceptron_tagger')  # For POS tagging
nltk.download('maxent_ne_chunker')  # For NER
nltk.download('words')  # For NER
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\nms31\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\nms31\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\nms31\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\nms31\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\nms31\AppData\Roaming\nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\nms31\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
```

```
Out[37]: True
```

Step 2: Tokenization

Instructions:

1. Use the `word_tokenize` function to split a sample text into words.
2. Use `sent_tokenize` to split a sample text into sentences.

```
In [39]: from nltk.tokenize import word_tokenize, sent_tokenize

# Sample text
text = "Natural Language Processing (NLP) is a fascinating field of Artificial Intelligence. It enables machines to u

# Word Tokenization
words = word_tokenize(text)
print("Word Tokens:", words)

# Sentence Tokenization
```

```
sentences = sent_tokenize(text)
print("Sentence Tokens:", sentences)
```

```
Word Tokens: ['Natural', 'Language', 'Processing', '(', 'NLP', ')', 'is', 'a', 'fascinating', 'field', 'of', 'Artific
ial', 'Intelligence', '.', 'It', 'enables', 'machines', 'to', 'understand', 'human', 'language', '.']
Sentence Tokens: ['Natural Language Processing (NLP) is a fascinating field of Artificial Intelligence.', 'It enables
machines to understand human language.']
```

Step 3: Stopword Removal

Instructions:

1. Filter out common words (stopwords) using the NLTK `stopwords` corpus.

In [41]:
```
from nltk.corpus import stopwords

# Define stop words
stop_words = set(stopwords.words('english'))

# Remove stopwords
filtered_words = [word for word in words if word.lower() not in stop_words]
print("Filtered Words:", filtered_words)
```

```
Filtered Words: ['Natural', 'Language', 'Processing', '(', 'NLP', ')', 'fascinating', 'field', 'Artificial', 'Intelli
gence', '.', 'enables', 'machines', 'understand', 'human', 'language', '.']
```

Step 4: Stemming

Instructions:

1. Reduce each word to its root form using the Porter Stemmer.

In [43]:
```
from nltk.stem import PorterStemmer

# Initialize the stemmer
stemmer = PorterStemmer()

# Apply stemming
stems = [stemmer.stem(word) for word in filtered_words]
print("Stemmed Words:", stems)
```

```
Stemmed Words: ['natur', 'languag', 'process', '(', 'nlp', ')', 'fascin', 'field', 'artifici', 'intellig', '.', 'enab
l', 'machin', 'understand', 'human', 'languag', '.']
```

Step 5: Lemmatization

Instructions:

1. Reduce words to their base or dictionary form using the WordNet Lemmatizer.

In [45]:
```
from nltk.stem import WordNetLemmatizer

# Initialize the lemmatizer
lemmatizer = WordNetLemmatizer()

# Apply lemmatization
lemmas = [lemmatizer.lemmatize(word) for word in filtered_words]
print("Lemmatized Words:", lemmas)
```

```
Lemmatized Words: ['Natural', 'Language', 'Processing', '(', 'NLP', ')', 'fascinating', 'field', 'Artificial', 'Intel
ligence', '.', 'enables', 'machine', 'understand', 'human', 'language', '.']
```

Step 6: POS Tagging

Instructions:

1. Identify the grammatical role (e.g., noun, verb) of each word using the `pos_tag` function.

In [47]:
```
from nltk import pos_tag

# Apply POS tagging
pos_tags = pos_tag(filtered_words)
print("POS Tags:", pos_tags)
```

```
POS Tags: [('Natural', 'JJ'), ('Language', 'NNP'), ('Processing', 'NNP'), ('(', '('), ('NLP', 'NNP'), (')', ')'), ('f
ascinating', 'VBG'), ('field', 'NN'), ('Artificial', 'NNP'), ('Intelligence', 'NNP'), ('.', '.'), ('enables', 'NNS'),
('machines', 'NNS'), ('understand', 'VBP'), ('human', 'JJ'), ('language', 'NN'), ('.', '.')]
```

Step 7: Named Entity Recognition (NER)

Instructions:

1. Extract named entities (e.g., names, organizations) from the text using `ne_chunk`.

In [49]:
```python
from nltk import ne_chunk

# Apply NER
ner = ne_chunk(pos_tag(word_tokenize(text)))
print("Named Entities:")
print(ner)
```

```
Named Entities:
(S
  Natural/JJ
  Language/NNP
  Processing/NNP
  (/(
  (ORGANIZATION NLP/NNP)
  )/)
  is/VBZ
  a/DT
  fascinating/JJ
  field/NN
  of/IN
  (ORGANIZATION Artificial/JJ Intelligence/NNP)
  ./.
  It/PRP
  enables/VBZ
  machines/NNS
  to/TO
  understand/VB
  human/JJ
  language/NN
  ./.)
```

Step 8: Practice Sentiment Analysis

Instructions:

1. Use the above techniques to analyze sentiment in a given text.
2. Identify positive, negative, or neutral sentiment using tokenization and stopword removal as preprocessing steps.

In [51]:
```
# Define a simple example for sentiment analysis
positive_words = ["fascinating", "enable", "understand"]
negative_words = ["problem", "complex"]

# Count positive and negative words
positive_count = sum([1 for word in filtered_words if word.lower() in positive_words])
negative_count = sum([1 for word in filtered_words if word.lower() in negative_words])

# Determine sentiment
if positive_count > negative_count:
    print("The sentiment is Positive.")
elif negative_count > positive_count:
    print("The sentiment is Negative.")
else:
    print("The sentiment is Neutral.")
```

```
The sentiment is Positive.
```