Text Analysis and Natural Language Processing (NLP) with NLTK

Natural Language Toolkit (NLTK) is a popular Python library for processing and analyzing human language data. It provides tools for text preprocessing, tokenization, stemming, lemmatization, sentiment analysis, and more.

---

1. Install and Import NLTK

Install NLTK and download required datasets.

bash
Copy code
```
pip install nltk
```

Download essential NLTK datasets:

python
Copy code
```
import nltk
nltk.download('punkt')  # Tokenizer
nltk.download('stopwords')  # Stopwords
nltk.download('wordnet')  # WordNet Lemmatizer
nltk.download('averaged_perceptron_tagger')  # POS tagging
nltk.download('vader_lexicon')  # Sentiment analysis
nltk.download('nps_chat')  # Example corpus
```

---

2. Tokenization

Tokenization splits text into sentences or words.

python
Copy code
```
from nltk.tokenize import word_tokenize, sent_tokenize

# Example text
text = "Natural Language Processing is a fascinating field. Let's
explore it with NLTK!"

# Sentence tokenization
sentences = sent_tokenize(text)
```

```
print("Sentences:", sentences)

# Word tokenization
words = word_tokenize(text)
print("Words:", words)
```

---

3. Removing Stopwords

Stopwords are common words (e.g., "the", "and", "is") that are often removed in text analysis.

python
Copy code
```
from nltk.corpus import stopwords

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

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

---

4. Stemming and Lemmatization

- **Stemming**: Reduces words to their root form (e.g., "running" → "run").
- **Lemmatization**: Reduces words to their base form, considering meaning (e.g., "better" → "good").

Stemming
python
Copy code
```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
stemmed_words = [stemmer.stem(word) for word in filtered_words]
print("Stemmed Words:", stemmed_words)
```

Lemmatization
python

Copy code
```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
lemmatized_words = [lemmatizer.lemmatize(word) for word in
filtered_words]
print("Lemmatized Words:", lemmatized_words)
```

---

5. Part-of-Speech (POS) Tagging

Assign grammatical tags (e.g., noun, verb) to words.

python
Copy code
```
from nltk import pos_tag

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

---

6. Named Entity Recognition (NER)

Identify proper nouns and entities in text.

python
Copy code
```
from nltk.chunk import ne_chunk

# Perform named entity recognition
ner_tree = ne_chunk(pos_tags)
print("Named Entities:", ner_tree)
```

---

7. Text Frequency Analysis

Frequency Distribution
python
Copy code
```
from nltk.probability import FreqDist
```

```
# Frequency distribution of words
fdist = FreqDist(filtered_words)
print("Most Common Words:", fdist.most_common(5))

# Plot the frequency distribution
fdist.plot(10, title="Word Frequency Distribution")
```

N-grams Analysis

N-grams are sequences of `n` consecutive words.

python
Copy code

```
from nltk.util import ngrams

# Generate bigrams
bigrams = list(ngrams(filtered_words, 2))
print("Bigrams:", bigrams)
```

---

8. Sentiment Analysis

Use NLTK's VADER (Valence Aware Dictionary and sEntiment Reasoner) for sentiment analysis.

python
Copy code

```
from nltk.sentiment import SentimentIntensityAnalyzer

# Initialize sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Analyze sentiment
sentiment = sia.polarity_scores(text)
print("Sentiment Analysis:", sentiment)
```

---

9. WordCloud Visualization

Visualize frequent words using the WordCloud library.

bash
Copy code

```
pip install wordcloud
```

python
Copy code

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Generate a word cloud
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(' '.join(filtered_words))

# Plot the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

---

10. Working with Corpora

NLTK provides access to various text corpora for experimentation.

Example: Chat Corpus

python
Copy code

```
from nltk.corpus import nps_chat

# Load chat corpus
posts = nps_chat.posts()
print("First Chat Post:", posts[0])
```

Example: Reuters Corpus

python
Copy code

```
nltk.download('reuters')
from nltk.corpus import reuters

# Get a list of file IDs and categories
print("Categories:", reuters.categories()[:5])
print("File IDs:", reuters.fileids()[:5])

# Access text
```

```
print("Text Example:", reuters.raw(reuters.fileids()[0])[:500])
```

---

11. Advanced NLP with NLTK

- **TF-IDF (Term Frequency-Inverse Document Frequency)**: Measure word importance in a document relative to a corpus.
- **Topic Modeling**: Use libraries like Gensim with NLTK-preprocessed text for advanced topic modeling.
- **Custom Tokenizers**: Define custom rules for splitting text into tokens.