---

## Text Preprocessing and Sentiment Analysis Documentation

### Step 1: Text Preprocessing

Text preprocessing involves cleaning and preparing the text data for analysis. In our case, we're working with Jane Austen's texts. Here are the steps we've taken for text preprocessing:

1. \*\*Load Text Data\*\*: We've loaded the Jane Austen texts from NLTK's Gutenberg corpus.

```python

import nltk

from nltk.corpus import gutenberg

# Load Jane Austen's texts from NLTK's Gutenberg corpus

emma = gutenberg.raw('austen-emma.txt')

pride\_and\_prejudice = gutenberg.raw('austen-persuasion.txt')

sense\_and\_sensibility = gutenberg.raw('austen-sense.txt')

```

2. \*\*Tokenization\*\*: We've tokenized the text data, splitting it into individual words or tokens.

```python

# Tokenize the text

emma\_tokens = nltk.word\_tokenize(emma)

pride\_and\_prejudice\_tokens = nltk.word\_tokenize(pride\_and\_prejudice)

sense\_and\_sensibility\_tokens = nltk.word\_tokenize(sense\_and\_sensibility)

```

### Step 2: Sentiment Analysis

Sentiment analysis involves analyzing the sentiment or emotion expressed in the text. We've used lexicon-based approaches for sentiment analysis.

1. \*\*Lexicon Initialization\*\*: We've initialized lexicons such as AFINN and Bing Liu for sentiment analysis.

```python

from afinn import Afinn

from nltk.corpus import opinion\_lexicon

# Initialize AFINN

afinn = Afinn()

# Load Bing Liu Lexicon

positive\_words = set(opinion\_lexicon.positive())

negative\_words = set(opinion\_lexicon.negative())

```

2. \*\*Calculating Sentiment Scores\*\*: We've calculated sentiment scores for each word in the text data using the chosen lexicons.

```python

# Calculate sentiment scores for each word using AFINN

afinn\_scores = [afinn.score(token) for token in emma\_tokens]

# Calculate sentiment scores for each word using Bing Liu Lexicon

bing\_liu\_scores = [1 if word in positive\_words else -1 if word in negative\_words else 0 for word in emma\_tokens]

```

3. \*\*Aggregating Sentiment Scores\*\*: We've aggregated sentiment scores for each document by summing up the scores of all words in the document.

```python

# Aggregate sentiment scores for each document

afinn\_document\_score = sum(afinn\_scores)

bing\_liu\_document\_score = sum(bing\_liu\_scores)

```

4. \*\*Classifying Sentiments\*\*: We've classified documents into positive or negative sentiments based on the aggregate sentiment scores.

```python

def classify\_sentiment(score):

if score > 0:

return "Positive"

elif score < 0:

return "Negative"

else:

return "Neutral"

# Classify sentiment for documents using AFINN

afinn\_sentiment = classify\_sentiment(afinn\_document\_score)

# Classify sentiment for documents using Bing Liu Lexicon

bing\_liu\_sentiment = classify\_sentiment(bing\_liu\_document\_score)

```

### Step 3: Visualization

Visualization helps in understanding the sentiment distribution within the text. We've created word clouds to visualize words associated with positive and negative sentiments.

```python

from wordcloud import WordCloud

import matplotlib.pyplot as plt

# Filter tokens based on sentiment scores (positive and negative)

positive\_tokens = [token for token, score in zip(emma\_tokens, afinn\_scores) if score > 0]

negative\_tokens = [token for token, score in zip(emma\_tokens, afinn\_scores) if score < 0]

# Join tokens into strings

positive\_text = " ".join(positive\_tokens)

negative\_text = " ".join(negative\_tokens)

# Create WordCloud for positive sentiment

positive\_wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(positive\_text)

# Create WordCloud for negative sentiment

negative\_wordcloud = WordCloud(width=800, height=400, background\_color='black', colormap='Reds').generate(negative\_text)

# Plot WordClouds

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.imshow(positive\_wordcloud, interpolation='bilinear')

plt.title('Positive Sentiment Word Cloud')

plt.axis('off')

plt.subplot(1, 2, 2)

plt.imshow(negative\_wordcloud, interpolation='bilinear')

plt.title('Negative Sentiment Word Cloud')

plt.axis('off')

plt.show()

```

---

This documentation summarizes all the steps involved in text preprocessing, sentiment analysis, and visualization, along with code snippets and explanations for each step.