Natural Language Processing using Python Programming

Notebook 03.2: Using Real-World Datasets (Pandas and EDA)

Python 3.8+ NLTK Latest SpaCy Latest Pandas Latest License MIT

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

Learning Objectives:

- Load and process real-world text datasets using Pandas DataFrames
- Integrate preprocessing pipelines with large-scale data operations
- Conduct comprehensive Exploratory Data Analysis (EDA) on text data
- Visualize text patterns using word clouds and statistical plots
- Establish production-ready data workflows for NLP projects

- In the real world, text data rarely comes in a clean NLTK corpus format.
- It often arrives as CSV, JSON, or from a database.
- This notebook focuses on the industry standard workflow: loading data into **Pandas DataFrames**, applying our preprocessing pipeline, and conducting basic **Exploratory Data Analysis (EDA)**.

1. Data Loading and Initial Inspection

- We will load a simulated IMDB movie review dataset from the `data/raw/` directory.
- **Pandas** is the primary tool for this.

In [1]:
```python
# Script to load and inspect the IMDB movie reviews dataset
# Import necessary libraries
import pandas as pd

# Define the path to the raw data file
FILE_PATH = '../../data/raw/imdb_movie_reviews.csv'

# Load the dataset
try:
    df = pd.read_csv(FILE_PATH)
    print("Data loaded successfully.")
    print("\nInitial DataFrame Head:")
    print(df.head())
    print("\nDataFrame Info:")
```

```python
    df.info()
except FileNotFoundError:
    print(f"ERROR: File not found at {FILE_PATH}. Please ensure 'imdb_movie_review
    df = None
```

```
Data loaded successfully.

Initial DataFrame Head:
                                              review sentiment
0  The film was absolutely stunning! Great acting...  positive
1  Worst movie I've seen all year. Predictable, b...  negative
2  It was okay, not great, not terrible. Just a s...   neutral
3  I can't believe they spent $200M on this. What...  negative
4  Truly a masterpiece of modern cinema. Don't mi...  positive

DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review     5 non-null      object
 1   sentiment  5 non-null      object
dtypes: object(2)
memory usage: 212.0+ bytes
```

---

2. Text Cleaning and Preprocessing Integration

- We'll now integrate the preprocessing steps we learned in Chapter 2.1 by defining a cleaner function and applying it to the entire DataFrame column.

In [2]:

```python
# Import SpaCy and re (regular expressions) libraries for text processing
import spacy
import re

# Load SpaCy model (only if df loaded successfully)
if df is not None:
    nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])

    def clean_and_lemmatize(text):
        """Lowercasing, Punctuation removal, Lemmatization, and Stopword removal.'
        if pd.isna(text):
            return ""

        # 1. Lowercase and remove noise
        text_lower = str(text).lower()
        text_clean = re.sub(r'[^a-z\s]', '', text_lower)

        # 2. Process with SpaCy for Lemmatization
        doc = nlp(text_clean)

        # 3. Filter stopwords and non-meaningful tokens, and get the lemma
        tokens = [token.lemma_ for token in doc
                  if not token.is_stop and
                  not token.is_punct and
                  not token.is_space]
```

```
        return " ".join(tokens)

    # Apply the cleaning function to the 'review' column
    print("Applying preprocessing to the 'review' column...")
    df['cleaned_review'] = df['review'].apply(clean_and_lemmatize)

    print("\nDataFrame Head after Cleaning:")
    print(df[['review', 'cleaned_review']].head())
```

```
Applying preprocessing to the 'review' column...

DataFrame Head after Cleaning:
                                              review  \
0  The film was absolutely stunning! Great acting...
1  Worst movie I've seen all year. Predictable, b...
2  It was okay, not great, not terrible. Just a s...
3  I can't believe they spent $200M on this. What...
4  Truly a masterpiece of modern cinema. Don't mi...

                                      cleaned_review
0  film absolutely stunning great acting fantasti...
1  bad movie ve see year predictable boring sound...
2        okay great terrible solid bmovie experience
3              not believe spend m waste talent time
4           truly masterpiece modern cinema not miss
```

---

3. Exploratory Data Analysis (EDA)

- EDA in NLP involves analyzing characteristics of the text before modeling, such as the length of reviews, the distribution of sentiment, and the most common words.

3.1 Analyzing Review Length

- Length (word or character count) can be an important feature.
- Longer reviews might be more expressive or, conversely, spam.

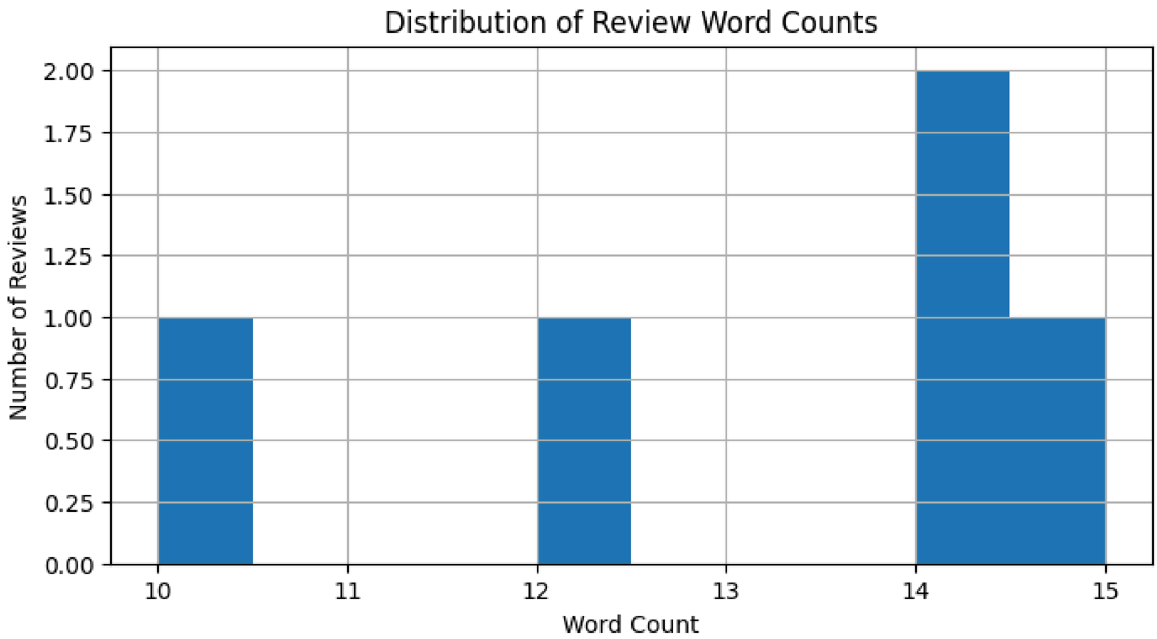
In [3]:
```
if df is not None:
    # Calculate word count for the original review
    df['word_count'] = df['review'].apply(lambda x: len(str(x).split()))

    print("Review Word Count Statistics:")
    print(df['word_count'].describe())

    # Visualizing the distribution (requires matplotlib)
    import matplotlib.pyplot as plt

    # Plotting the distribution of review lengths using word count
    # Histogram of word counts
    plt.figure(figsize=(8, 4))
    df['word_count'].hist(bins=10) # Reduced bins for clarity
    plt.title('Distribution of Review Word Counts')
    plt.xlabel('Word Count')
    plt.ylabel('Number of Reviews')
    plt.show()
```

```
Review Word Count Statistics:
count     5.0
mean     13.0
std       2.0
min      10.0
25%      12.0
50%      14.0
75%      14.0
max      15.0
Name: word_count, dtype: float64
```



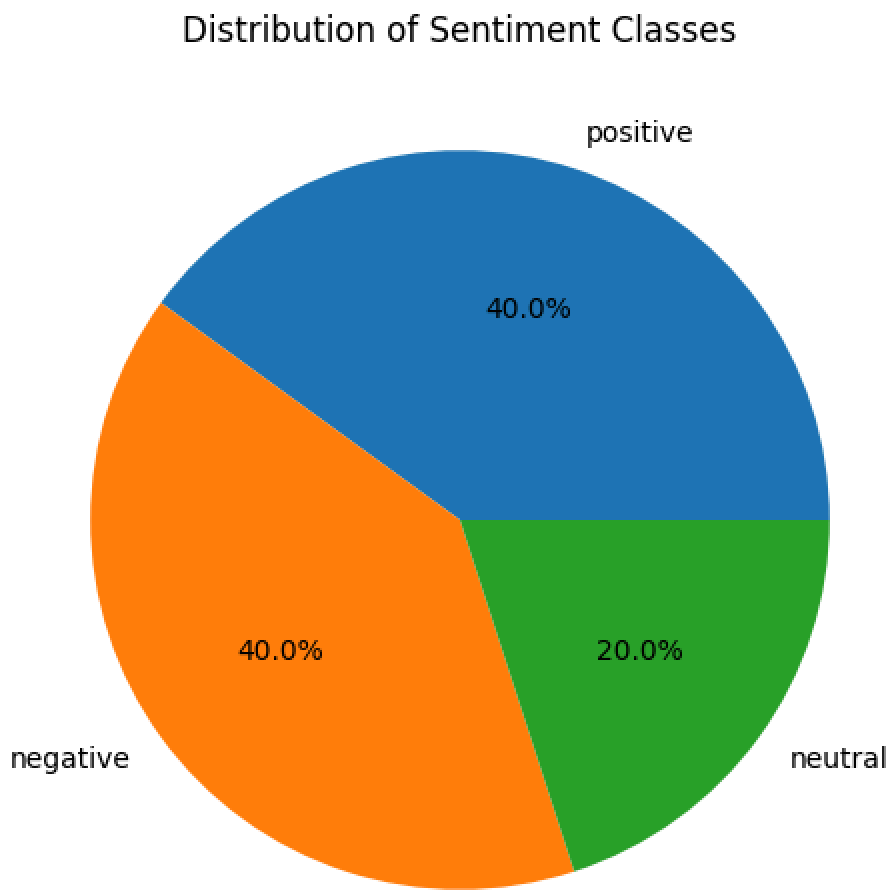
3.2 Analyzing Target Variable Distribution

- Understanding the distribution of the target variable (`sentiment`) is critical for classification.
- Imbalanced data (e.g., far more positive than negative reviews) requires special handling.

In [4]:
```
if df is not None:
    sentiment_counts = df['sentiment'].value_counts()
    print("Sentiment Distribution:")
    print(sentiment_counts)

    # Plotting the distribution using a pie chart
    plt.figure(figsize=(6, 6))
    sentiment_counts.plot(kind='pie', autopct='%1.1f%%')
    plt.title('Distribution of Sentiment Classes')
    plt.ylabel('')
    plt.show()
```

```
Sentiment Distribution:
sentiment
positive    2
negative    2
neutral     1
Name: count, dtype: int64
```



3.3 Visualizing Word Frequency (Word Clouds)

- A **Word Cloud** is a visualization that gives greater prominence to words that appear more frequently in the source text, providing a quick visual summary of the corpus's vocabulary.

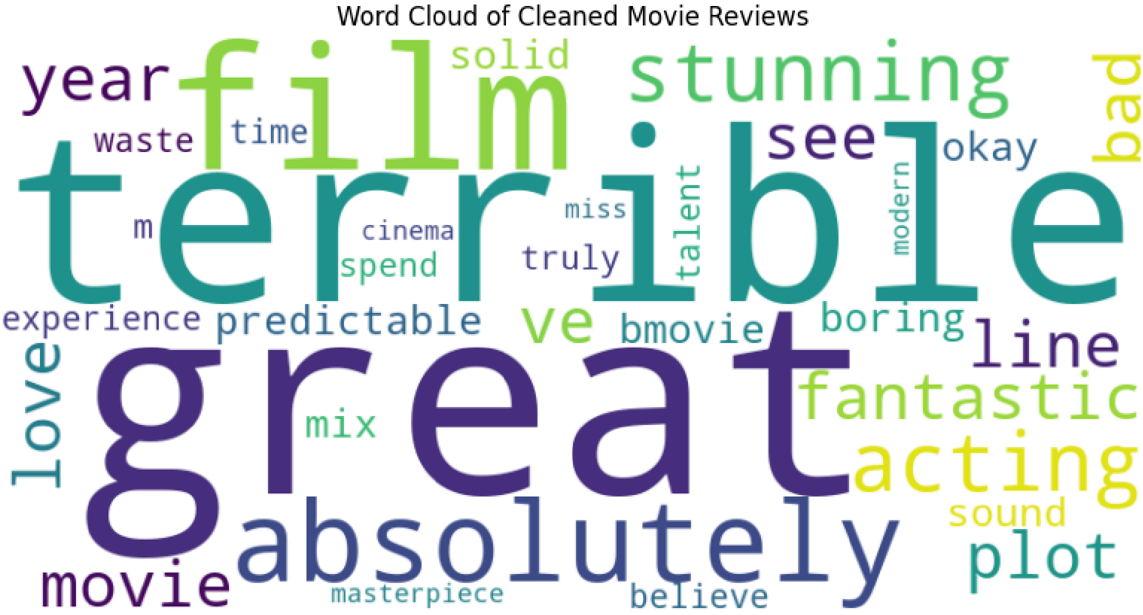
In [ ]:

```
if df is not None:
    # Import WordCloud library
    from wordcloud import WordCloud

    # Combine all cleaned text into one large string
    all_text = ' '.join(df['cleaned_review'].dropna())

    # Generate a word cloud image
    wordcloud = WordCloud(width=800, height=400, background_color='white').generat

    # Display the generated image:
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear') # imshow() for displaying imag
    plt.axis('off')
    plt.title('Word Cloud of Cleaned Movie Reviews')
    plt.show()
```



4. Saving the Processed Data

- A best practice in data science is to save the clean, processed data.
- This prevents us from having to run the time-consuming preprocessing steps every time we start modeling.

In [8]:
```
if df is not None:
    df.to_csv('../../data/processed/processed_reviews.csv', index=False)
    print("\nProcessed data saved to: data/processed/processed_reviews.csv")
    print("This clean file is now ready for feature extraction and modeling (Chapt
```

```
Processed data saved to: data/processed/processed_reviews.csv
This clean file is now ready for feature extraction and modeling (Chapters 6-8).
```

5. Summary and Next Steps

- We successfully loaded real-world data with Pandas, integrated a sophisticated cleaning pipeline, and performed key EDA steps using visualization.
- In **Chapter 4**, we move deeper into language structure by exploring **Part-of-Speech (POS) Tagging** and **Dependency Parsing** to understand the grammatical roles and relationships between words.

Key Takeaways

- **Real-World Data Processing:** We successfully loaded and processed industry-standard text datasets using Pandas, moving beyond static corpora to dynamic data workflows.
- **Integrated Preprocessing:** We applied our comprehensive text preprocessing pipeline to large datasets, demonstrating scalable text cleaning and normalization techniques.

- **Exploratory Data Analysis:** We conducted thorough EDA including sentiment distribution analysis, word count statistics, and visual text exploration through word clouds.
- **Production Workflow:** We established best practices for saving processed data and creating reproducible data science workflows.

Next Notebook Preview

- Now that we can process real-world datasets, we're ready to dive deeper into **linguistic structure and grammar**.
- The next chapter will explore **Part-of-Speech (POS) Tagging and Dependency Parsing** to understand grammatical roles and word relationships in text.

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

Built with ❤️ for the Python community