Natural Language Processing using Python Programming

Notebook 07.1: Sentiment Analysis with NLTK (Lexicon-Based Approach)

Python 3.8+ NLTK Latest SpaCy Latest Scikit-learn Latest License MIT

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

Learning Objectives:

- Master lexicon-based sentiment analysis using NLTK's VADER analyzer
- Understand the fundamental approaches to sentiment analysis (lexicon vs ML)
- Implement VADER for real-time sentiment scoring without training data
- Interpret compound scores and sentiment classification thresholds
- Analyze sentiment patterns in real-world movie review datasets

---

- **Sentiment Analysis** (or opinion mining) is the process of computationally identifying and categorizing opinions expressed in a piece of text, especially to determine the writer's attitude toward a particular topic or product.
- It is typically categorized as positive, negative, or neutral.
- There are two main approaches:
  1. **Lexicon-Based:** Uses a dictionary of words (a lexicon) pre-labeled with sentiment scores.
  2. **Machine Learning:** Trains a classifier (like Logistic Regression) on labeled data (e.g., Chapter 8).
- This notebook focuses on the fast and robust **Lexicon-Based** approach using **NLTK's VADER**.

1. Setting up: Libraries and VADER

- **VADER** (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analyzer specifically attuned to sentiments expressed in social media.
- It considers punctuation, capitalization, and use of modifiers (like 'not').

In [1]:
```
# Import necessary libraries
import nltk
```

```python
import pandas as pd
import matplotlib.pyplot as plt

# VADER requires the 'vader_lexicon' resource
nltk.download('vader_lexicon', quiet=True)

# SentimentIntensityAnalyzer is use for sentiment analysis
# Initialize VADER Sentiment Analyzer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()

print("VADER Analyzer initialized.")
```

```
VADER Analyzer initialized.
```

---

2. VADER Output and Interpretation

- VADER produces a dictionary of four scores for any given text:

    1. `neg` **(Negative):** Proportion of negative sentiment.
    2. `neu` **(Neutral):** Proportion of neutral sentiment.
    3. `pos` **(Positive):** Proportion of positive sentiment.
    4. `compound` **(Compound Score):** A normalized, aggregated score (-1.0 to +1.0). This is the most common single metric used to determine overall sentiment.

In [2]:

```python
# Sample texts for sentiment analysis
text_positive = "This product is absolutely amazing! I highly recommend it."
text_negative = "Worst customer experience; everything was delayed and terrible."
text_neutral = "The meeting started on time and ended as scheduled."
text_sarcastic = "The service was absolutely great... NOT!" # VADER handles negat

texts = [text_positive, text_negative, text_neutral, text_sarcastic]
labels = ['Positive', 'Negative', 'Neutral', 'Sarcastic']

# zip() pairs each label with its corresponding text
# polarity_scores() returns a dictionary of scores
for label, text in zip(labels, texts):
    scores = analyzer.polarity_scores(text)
    print(f"--- {label} ---")
    print(f"Text: {text}")
    print(f"Scores: {scores}")
    print(f"Compound Score: {scores['compound']:.4f}\n")
```

```
--- Positive ---
Text: This product is absolutely amazing! I highly recommend it.
Scores: {'neg': 0.0, 'neu': 0.446, 'pos': 0.554, 'compound': 0.8147}
Compound Score: 0.8147

--- Negative ---
Text: Worst customer experience; everything was delayed and terrible.
Scores: {'neg': 0.645, 'neu': 0.355, 'pos': 0.0, 'compound': -0.8442}
Compound Score: -0.8442

--- Neutral ---
Text: The meeting started on time and ended as scheduled.
Scores: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Compound Score: 0.0000

--- Sarcastic ---
Text: The service was absolutely great... NOT!
Scores: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Compound Score: 0.0000
```

> **VADER Rule:** Typically, a Compound Score is interpreted as:

- **Positive:** $\text{Compound} \geq 0.05$
- **Negative:** $\text{Compound} \leq -0.05$
- **Neutral:** $-0.05 < \text{Compound} < 0.05$

---

3. Analyzing a Real-World DataFrame

- We will apply VADER to the original, unprocessed movie reviews from Chapter 3.2.
- Lexicon-based approaches often work best on *unprocessed* text because they rely on capitalization, punctuation, and emojis for intensity.

In [3]:

```python
# Load the original data (since VADER works best on raw text)
FILE_PATH = '../../data/raw/imdb_movie_reviews.csv'

try:
    df_raw = pd.read_csv(FILE_PATH)

    # Apply VADER to the 'review' column
    df_raw['vader_compound'] = df_raw['review'].apply(lambda x: analyzer.polarity_

    # Map compound score to a simple categorical prediction
    def categorize_vader(score):
        if score >= 0.05:
            return 'positive'
        elif score <= -0.05:
            return 'negative'
        else:
            return 'neutral'

    df_raw['vader_prediction'] = df_raw['vader_compound'].apply(categorize_vader)
```

```
    print("VADER Analysis Results:")
    print(df_raw[['review', 'sentiment', 'vader_compound', 'vader_prediction']].he

except FileNotFoundError:
    print(f"ERROR: Raw data not found at {FILE_PATH}. Cannot run analysis.")
```

```
VADER Analysis Results:
                                              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

   vader_compound vader_prediction
0          0.9465         positive
1         -0.8591         negative
2         -0.6108         negative
3          0.0000          neutral
4          0.8148         positive
```

Visualizing VADER Predictions

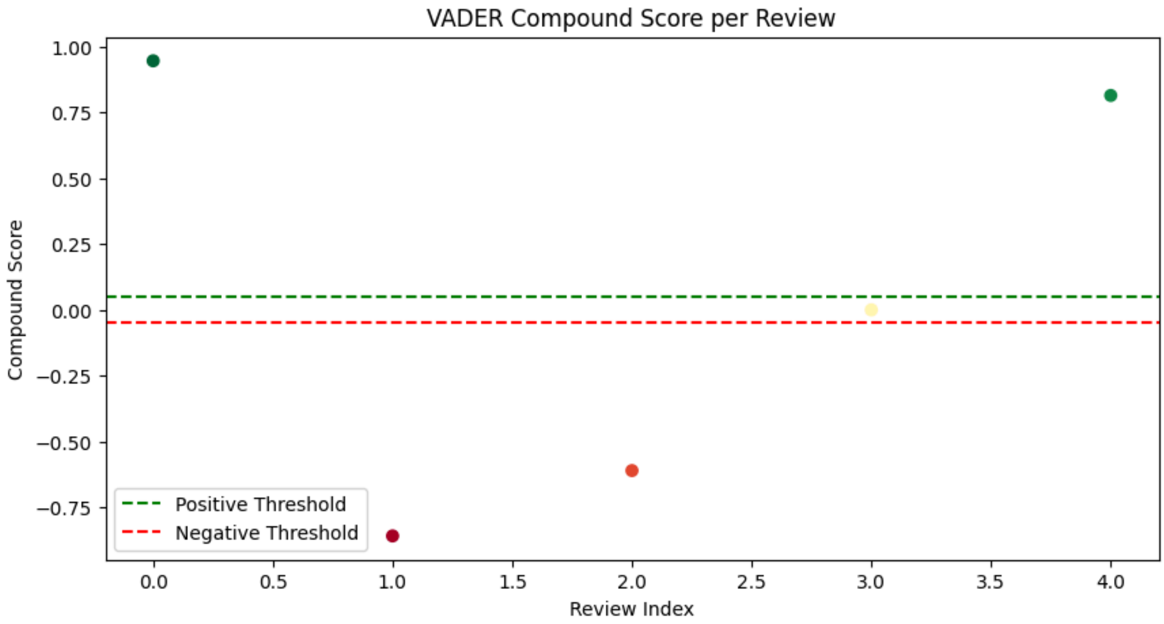
- We can visualize how VADER's predictions align with the original human-labeled sentiment (`sentiment`).

In [4]:
```
# Visualizing VADER Predictions
# locals() : Check if df_raw exists
if 'df_raw' in locals():
    # Create a simple cross-tabulation (Confusion Matrix conceptual)
    crosstab = pd.crosstab(df_raw['sentiment'], df_raw['vader_prediction'])

    print("\nHuman Label vs. VADER Prediction:")
    print(crosstab)

    # Plotting the compound scores
    plt.figure(figsize=(10, 5))
    plt.scatter(df_raw.index, df_raw['vader_compound'], c=df_raw['vader_compound']
    plt.axhline(y=0.05, color='g', linestyle='--', label='Positive Threshold')
    plt.axhline(y=-0.05, color='r', linestyle='--', label='Negative Threshold')
    plt.title('VADER Compound Score per Review')
    plt.xlabel('Review Index')
    plt.ylabel('Compound Score')
    plt.legend()
    plt.show()
```

```
Human Label vs. VADER Prediction:
vader_prediction  negative  neutral  positive
sentiment
negative                 1        1         0
neutral                  1        0         0
positive                 0        0         2
```



4. Summary and Next Steps

- VADER is a fantastic tool for **quick sentiment scoring**, especially on social media text where rules for slang and emojis are helpful.
- It requires **no training** but is limited by its fixed lexicon.
- For higher accuracy and customization, we need a **Machine Learning approach**.
- In the next notebook (**7.2**), we will pivot to the ML approach, using the features we created in Chapter 6 to train a classifier and achieve superior, general-purpose sentiment analysis.

Key Takeaways

- **Lexicon-Based Mastery:** We successfully implemented sentiment analysis using NLTK's VADER, a powerful rule-based analyzer that requires no training data.
- **VADER Understanding:** We learned to interpret VADER's four-score system (neg, neu, pos, compound) and apply standard thresholds for sentiment classification.
- **Real-World Application:** We analyzed actual movie reviews, comparing VADER predictions with human-labeled sentiment to understand performance characteristics.
- **Approach Comparison:** We established the foundation for comparing lexicon-based methods with machine learning approaches in terms of speed, accuracy, and customization.

Next Notebook Preview

- With lexicon-based sentiment analysis mastered, we're ready to explore **machine learning approaches** for superior accuracy and customization.

- The next notebook will implement **ML-based sentiment classification**, using the vectorized features from Chapter 6 to train supervised learning models.

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