**Sentio - Sentiment Analysis Model**

**Explanation of the Sentiment Analysis Model and NLP Usage**

This code is a **Sentiment Analysis App** built using **Natural Language Processing (NLP)**. It analyzes user-provided text and categorizes the sentiment of the text into positive, negative, or neutral. The app uses two NLP techniques for sentiment analysis: **TextBlob** and **VADER Sentiment Analysis**. Here's a breakdown of how NLP is applied and the overall flow of the model:

**1. NLP Concepts Used:**

**A. TextBlob for Sentiment Analysis:**

* **TextBlob** is a simple library for processing textual data. It is used here to compute **polarity** and **subjectivity** scores of the input text.
  + **Polarity** measures the positivity or negativity of the text (range: -1 to 1). A positive value indicates positive sentiment, while a negative value indicates negative sentiment.
  + **Subjectivity** measures how subjective the text is (range: 0 to 1). A value closer to 1 indicates that the text is more subjective, while a value closer to 0 indicates objectivity.

The **TextBlob Sentiment** object provides these two metrics (polarity and subjectivity), which are used to categorize the overall sentiment of the text.

**B. VADER Sentiment Analysis:**

* **VADER (Valence Aware Dictionary and sEntiment Reasoner)** is a more advanced sentiment analysis tool based on a lexicon of words and their sentiment scores. It uses **lexicon-based heuristics** to classify tokens (individual words) into categories like positive, negative, or neutral.
  + The **VADER SentimentIntensityAnalyzer** computes the **compound score** for each token, which is a normalized value that gives an overall sentiment.
  + Based on the compound score:
    - Positive if the score is > 0.1
    - Negative if the score is <= -0.1
    - Neutral otherwise

**2. Workflow of the Sentiment Analysis Model:**

**A. Text Input & Submission (Streamlit UI):**

* The user enters some text into a text area and clicks the **Analyze** button to analyze the sentiment.

**B. Text Analysis with TextBlob:**

* **Sentiment Analysis with TextBlob**:
  + The TextBlob(raw\_text).sentiment function is used to compute the **polarity** and **subjectivity** of the entire text.
  + These values are displayed to the user, and an emoji is shown to indicate whether the overall sentiment is **positive**, **negative**, or **neutral**.

**Example:**

* + Positive text: "I love this product!" → Polarity: 0.8 (Positive)
  + Negative text: "I hate waiting in long lines." → Polarity: -0.6 (Negative)

**Dataframe Output:**

* + A DataFrame is created using the convert\_to\_df function, which displays the **polarity** and **subjectivity** values.

**C. Token-Level Sentiment Analysis with VADER:**

* **Token-Level Sentiment Analysis with VADER**:
  + The analyze\_token\_sentiment function uses VADER to break down the sentence into individual **tokens (words)**.
  + Each word's **compound score** is computed using SentimentIntensityAnalyzer's polarity\_scores(i)['compound'].
  + The token is classified as **positive**, **negative**, or **neutral** based on its score.

**Example:**

* + For the text "I love this product but hate the shipping", the function might detect:
    - love → Positive sentiment (score > 0.1)
    - hate → Negative sentiment (score <= -0.1)
    - this and but → Neutral sentiment (score between -0.1 and 0.1)

These classified tokens are displayed in the app under **Token Sentiment**.

**D. Visualization with Altair:**

* The app uses **Altair** to visualize the **polarity** and **subjectivity** scores in a bar chart.
* The bar chart plots the **polarity** and **subjectivity** as metrics, making it easy to compare the sentiment levels.

**3. Key Features of the Model:**

* **TextBlob** helps analyze **overall sentiment** of the entire text, providing a **general view** of the sentiment (positive, negative, neutral).
* **VADER** focuses on **token-level sentiment** to detect **positive**, **negative**, and **neutral** sentiments for individual words, providing a more granular view of the text.
* **Streamlit** makes it easy to build an interactive app for sentiment analysis with a simple UI to input text and view the results, visualizations, and detailed breakdowns.
* **Altair Visualization** helps in representing the sentiment metrics graphically for better understanding.

**4. Practical Use of NLP:**

* **TextBlob** and **VADER** are popular choices in sentiment analysis because they are fast, easy to use, and do not require any extensive training data (i.e., they are pre-trained models).
* This app is useful in applications where you want to analyze customer feedback, social media posts, reviews, or any form of text to determine public sentiment towards a product, service, or event.

**Conclusion:**

This code uses two main NLP techniques to classify sentiment:

1. **TextBlob** provides a simple, holistic analysis of the overall sentiment (positive/negative/neutral).
2. **VADER** analyzes individual words (tokens) for a more fine-grained understanding of sentiment in the text.

The integration of these two methods allows users to understand sentiment at both the **document level** and **token level**.