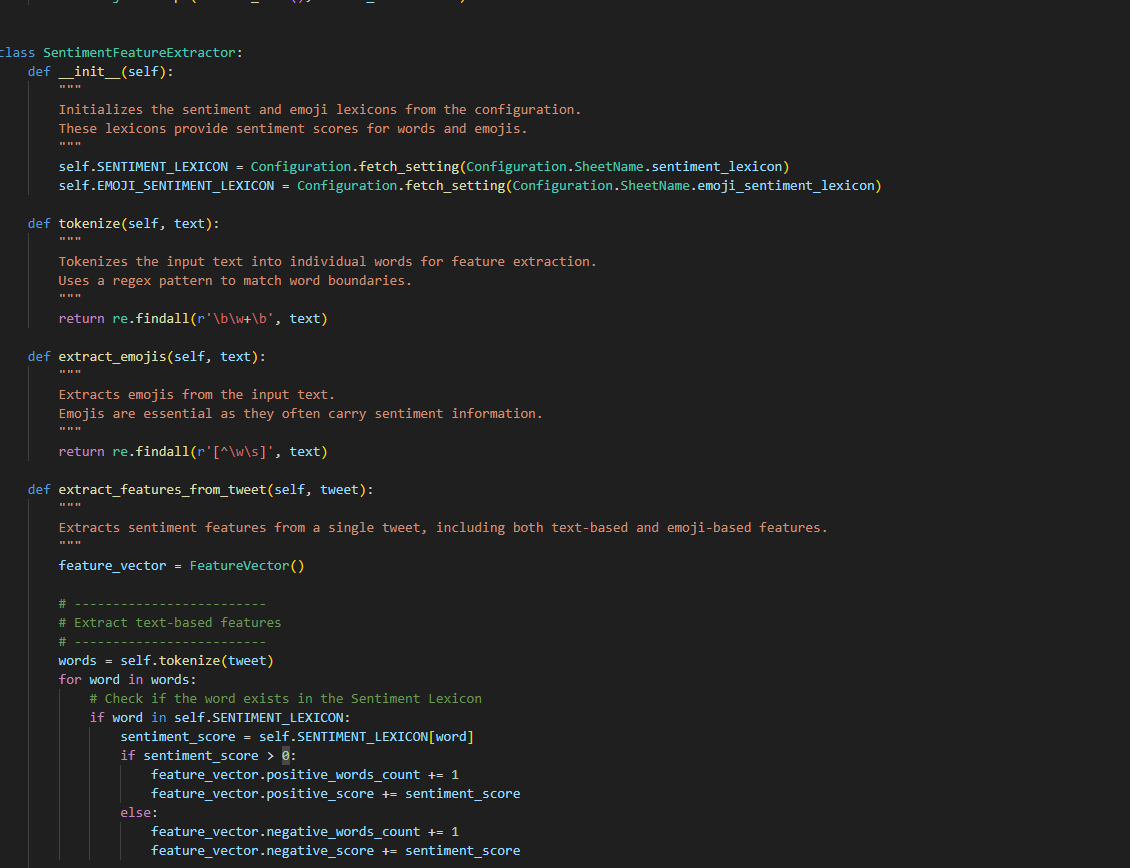


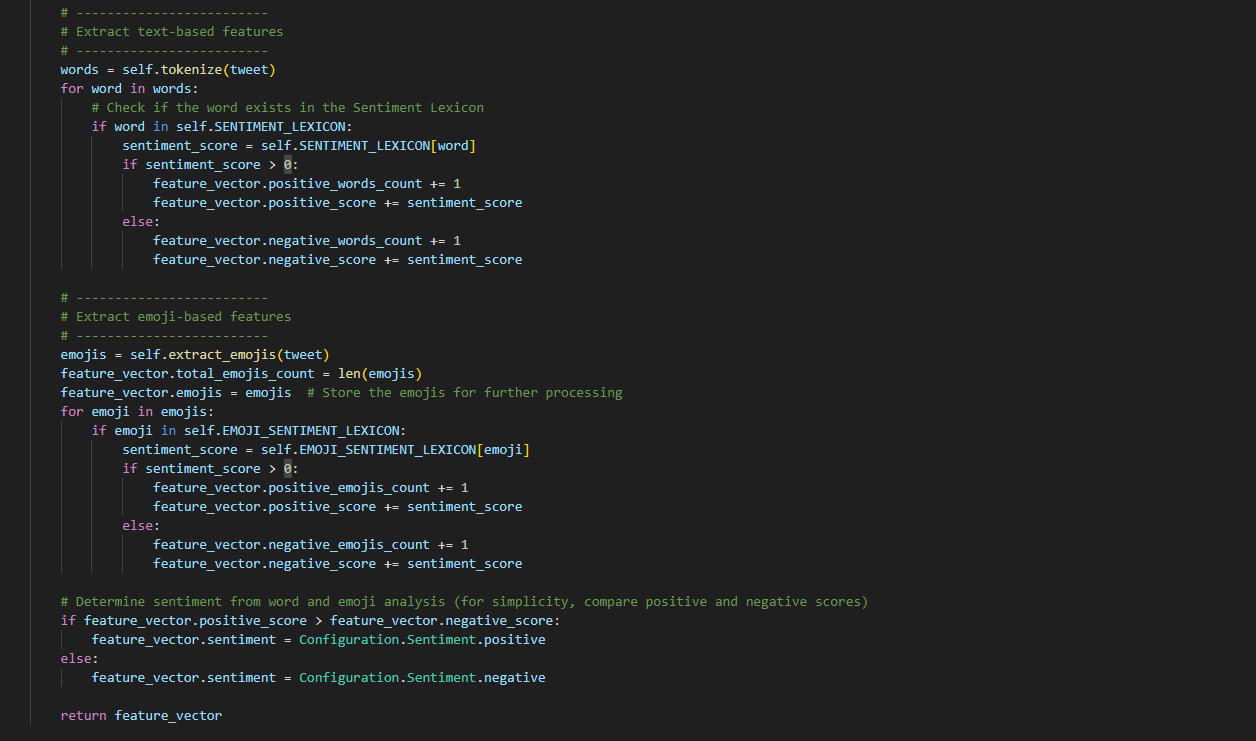
Initializes the feature vector to store counts and scores for sentiment analysis features.

This includes positive/negative words, emojis, and their sentiment scores.



Initializes the sentiment and emoji lexicons from the configuration.

These lexicons provide sentiment scores for words and emojis.



Extract text-based features

Check if the word exists in the Sentiment Lexicon

Extract emoji-based features

**Summary and Description of the Code:**

This code extracts sentiment-related features from a dataset of tweets. It analyzes both text-based and emoji-based sentiment to generate feature vectors that can be used for sentiment analysis.

**Key Components:**

1. **FeatureVector Class**:
   * This class represents the features extracted from each tweet. It stores counts and sentiment scores for positive/negative words, emojis, and overall sentiment.
   * It also provides methods to convert the feature vector into a dictionary and a compact string format for easy representation or logging.
2. **SentimentFeatureExtractor Class**:
   * This is the core class that performs the feature extraction process. It loads the sentiment lexicons for words and emojis from a configuration file and defines methods for:
     + **Tokenizing the tweet text**: Breaking the tweet into individual words.
     + **Extracting emojis from the tweet**: Identifying emoji characters.
     + **Extracting sentiment features**: Counting the occurrences of positive/negative words and emojis, and calculating the overall sentiment score.
3. **extract\_features\_from\_tweet() Method**:
   * This method processes a single tweet. It tokenizes the tweet text, checks each word and emoji against predefined sentiment lexicons, and updates counts and sentiment scores. It then determines whether the sentiment of the tweet is positive or negative based on the aggregated scores.
4. **Configuration Class**:
   * This class is assumed to handle the configuration and retrieval of settings like the sentiment lexicon, emoji lexicon, and the tweet dataset. It fetches necessary settings and provides them to the extractor for processing.
5. **Main Function**:
   * The main function simulates processing a list of tweets:
     + It fetches the tweets from a configuration.
     + It iterates over each tweet, uses the SentimentFeatureExtractor to extract sentiment features, and prints the features in two formats: detailed (JSON-like) and one-line (compact JSON).

**Workflow of the Code:**

1. **Initialization**:
   * The SentimentFeatureExtractor class initializes by loading sentiment lexicons (for words and emojis) from a configuration file.
2. **Feature Extraction**:
   * For each tweet:
     + The tweet is tokenized into words.
     + Each word is checked against the sentiment lexicon to extract sentiment scores (positive or negative).
     + Emojis are extracted from the tweet and checked against the emoji sentiment lexicon.
     + The feature vector is updated with counts and scores for words and emojis.
3. **Sentiment Determination**:
   * After extracting features, the sentiment of the tweet is determined by comparing the total positive and negative scores. If positive scores outweigh negative ones, the sentiment is labeled as positive, otherwise, it’s negative.
4. **Output**:
   * The program prints the feature vectors for each tweet in two formats:
     + **Detailed Format**: Shows all features in a readable, indented JSON format.
     + **One-Line Format**: Shows all features in a single-line JSON format for compact display.

**Overall Purpose:**

The code is designed to process and analyze tweets for sentiment using both textual and emoji data. By extracting features related to positive/negative words and emojis, it builds a detailed sentiment profile for each tweet, which could be further used for machine learning models or deeper sentiment analysis. The code can be easily extended to handle more complex features or to integrate with a larger sentiment analysis pipeline.