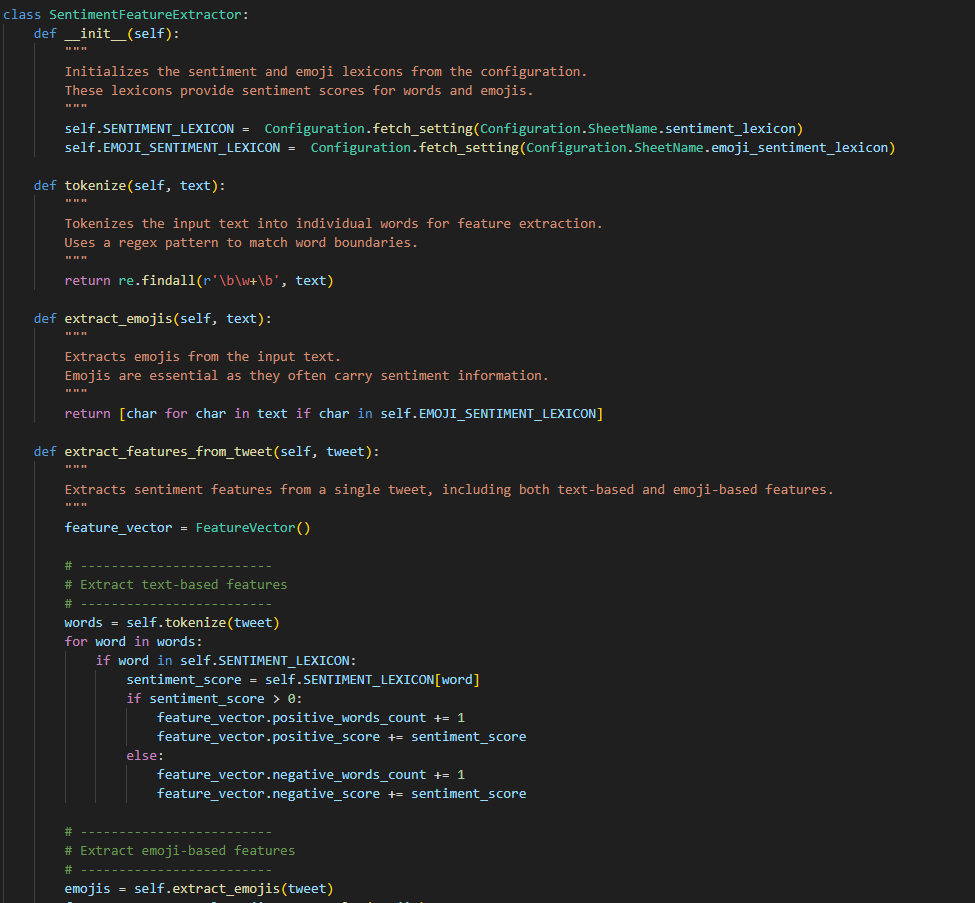


Initializes the feature vector to store counts and scores for sentiment analysis features.

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.

**Total Summary and Description of the Code**

The **Emo-SL Framework** for Emoji Sentiment Lexicon aims to enhance sentiment analysis by integrating emoji sentiment into traditional text-based sentiment analysis. It uses machine learning to combine both text and emoji features to classify sentiment accurately. The following describes the key algorithms, classes, and functions involved in this framework.

**Key Concepts:**

1. **Sentiment Lexicon**: The framework uses a sentiment lexicon for both words and emojis. Each word and emoji in the lexicon has a sentiment score (positive or negative) that helps classify the sentiment of a given tweet.
2. **Feature Extraction**: The framework extracts both text-based features (words) and emoji-based features to build a feature vector for each tweet.
3. **Sentiment Classification**: Based on the extracted features, the sentiment of a tweet is classified as either positive or negative. The final classification is determined by comparing the total positive and negative sentiment scores.

**Algorithm 3: Counting Emoji Occurrences**

This algorithm focuses on counting the occurrences of emojis in tweets based on their sentiment (positive or negative). The core idea is to:

* Iterate over the emojis in a tweet.
* Retrieve the sentiment score of each emoji.
* Track the count of positive and negative emojis separately.

The function count\_emojis receives a **FeatureVector** (containing extracted features) and uses the **emoji sentiment lexicon** to categorize emojis as positive or negative, while maintaining the counts of each emoji's occurrences.

**Main Classes and Functions:**

1. **FeatureVector Class**:
   * This class holds the features extracted from a tweet. It includes counts for positive/negative words and emojis, as well as the overall sentiment score.
   * **Methods**:
     + to\_dict: Converts the feature vector into a dictionary for easier representation.
     + \_\_str\_\_: Converts the feature vector into a human-readable JSON string for display.
     + to\_one\_line: Provides a compact JSON string representation for logging.
2. **SentimentFeatureExtractor Class**:
   * This class is responsible for extracting features from tweets. It handles the extraction of both text-based (word) and emoji-based features.
   * **Methods**:
     + tokenize: Tokenizes tweet text into words using a regular expression.
     + extract\_emojis: Extracts valid emojis from the tweet.
     + extract\_features\_from\_tweet: Main method for extracting features. It evaluates both words (using the sentiment lexicon) and emojis (using the emoji sentiment lexicon) to compute the sentiment scores.
3. **count\_emojis Function** (Algorithm 3):
   * This function takes the **FeatureVector** generated by the SentimentFeatureExtractor and the **emoji sentiment lexicon** to count the occurrences of positive and negative emojis in a tweet. It returns two dictionaries: one for positive emojis and one for negative emojis.
4. **main Function**:
   * The main() function is where the entire sentiment extraction process is demonstrated. It fetches a set of tweets, extracts features, counts the occurrences of positive and negative emojis using the count\_emojis function, and aggregates the results across all tweets. Finally, it prints the total counts of positive and negative emojis.

**Detailed Flow:**

1. **Tweet Processing**:
   * The framework processes each tweet in the dataset by extracting both word-based and emoji-based features.
   * For each tweet:
     + Words are tokenized and checked against the sentiment lexicon to calculate the positive and negative word counts and sentiment scores.
     + Emojis are extracted and their occurrences are counted, with their sentiment scores being applied to categorize them as positive or negative.
2. **Feature Vector Creation**:
   * The extracted features (word counts, emoji counts, sentiment scores) are stored in a **FeatureVector** for each tweet. This feature vector is used to represent the tweet's sentiment.
3. **Emoji Counting**:
   * The count\_emojis function categorizes emojis based on their sentiment scores (positive or negative) and maintains a count of each emoji's occurrences. The counts are aggregated across all tweets to provide a final summary of emoji sentiment in the dataset.
4. **Output**:
   * After processing all tweets, the final results (total positive and negative emoji counts) are displayed.

**Final Output Example:**

* The system will output the count of positive and negative emojis for each tweet processed, as well as a final aggregated summary of the total positive and negative emojis across all tweets. Example:
* Final Aggregated Emoji Counts:
* Total Positive Emojis Count: {'😊': 5, '😍': 3}
* Total Negative Emojis Count: {'😞': 2, '😠': 1}

**Conclusion:**

This framework enhances traditional sentiment analysis by integrating emojis into the process, recognizing their role in conveying sentiment. By extracting both text and emoji features, it offers a more nuanced understanding of the sentiment in social media data. The code is structured to first extract features, then classify sentiment, and finally count emoji occurrences based on their sentiment, providing a comprehensive sentiment analysis system.