1. Detecting Keywords and Triggering Sentiment Analysis

Before any sentiment is calculated, the `Handler.process_paragraph()` method checks if any keywords are present in the paragraph. If at least one keyword is found, then the paragraph's sentiment is computed.

2. Computing the Sentiment Score

Within the `SentimentAnalyzer` class, the method `analyze_sentiment(text)` performs the following steps:

`get_likelihood(text)`, first tokenizes the entire paragraph using the FinBERT tokenizer:

```
inputs = self.tokenizer(text, return_tensors="pt", padding=True,
truncation=True)

outputs = self.model(**inputs)
likelihoods = softmax(outputs.logits, dim=1)
```

Softmax is applied to the output logits to convert them into probabilities, representing the likelihood that the paragraph exhibits certain sentiments. The returned `likelihoods` tensor is assumed to have three elements corresponding to **Negative**, **Neutral**, and **Positive** sentiment.

b. Combining Likelihoods into a Single Sentiment Score

After the likelihoods are available, the code calculates a composite sentiment score using the formula:

```
sentiment_score = (likelihoods[NEUTRAL] + (likelihoods[POSITIVE] * 2)
+ (likelihoods[NEGATIVE] * 3)) - 2
```

The expression subtracts **2** from the total. This subtraction is done to re-center or calibrate the resulting score around a target value (0) so that the score reflects a relative sentiment intensity that takes both extremes (positive and negative) into account.

3. Determining the Sentiment Magnitude

Splitting into Sentences:
The paragraph is segmented into individual sentences with:

```
sentences = sent_tokenize(text)
```

VADER Scoring:
For each sentence, VADER computes sentiment scores. The code retrieves the compound score for each sentence

```
sentence_magnitude =
abs(self.sia.polarity_scores(sentence)['compound'])
```

Compound Score: This score, which normally ranges from -1 to 1, is taken in its absolute value. This means that whether the sentiment is positive or negative, its intensity is considered the same.

```
total_magnitude = sum(magnitudes)
```

All the absolute scores from each sentence are then added together. The result, `total_magnitude`, represents the overall intensity of emotion present in the paragraph.

4. Final Output for a Keyword-Containing Paragraph

The `analyze_sentiment(text)` method returns a tuple containing:

`Sentiment_score`, **`total_magnitude`**

As previously stated, sentiment_score is derived from finbert, total_magnitude is derived from VADER

After this, the `Handler.process_paragraph()` method proceeds to use this score for further processing (such as applying keyword-specific weights via `weight_sentiment`) and then storing the results via the `DataManager`.

Summary

For any paragraph containing a keyword:

- **FinBERT** is used to produce likelihoods for negative, neutral, and positive sentiments.

- These likelihoods are weighted (neutral at 1×, positive at 2×, negative at 3×) and then summed together, with 2 subtracted for calibration, resulting in a **sentiment score**.
- Simultaneously, **VADER** analyzes each sentence individually to compute an absolute sentiment intensity (magnitude), and these are summed for a **total magnitude**.
- The final results (score and magnitude) are then processed further (e.g., keyword weighting) and stored.