**CONCEPT:**

This class serves three purposes:

1. Estimate the sentiment for a string based on emotion words, booster words, emoticons and polarity changers
2. Allow you to save analysed data into positive, negative or neutral datasets
3. Identify if we have any phrase matches on previously analysed positive, negative and neutral phrases

Should there be any high quality phrase matches, it would take precedent over the sentiment analysis and return the phrase match rating instead.

**SENTIMENT ANALYSIS**

Strings are broken into tokenised arrays of single words. These words are analysed against TXT files that contain emotion words with ratings, emoticons with ratings, booster words with ratings and possible polarity changers.

A score is then calculated based on this analyse and this forms the "Sentiment analysis score".

**PHRASE ANALYSIS**

This function is key to identifying whether the phrase in questions can be compared to phrases that we have analysed and stored before. It uses Levenshtein distance to calculate distance between 4,5,6,7,8,9 and 10 word length phrases against the dataset we already have. We also make use of PHP's similar\_text to double verify proximity.

This means that the more phrases we have analysed previously improves the entire dataset and allows phrases to be more accurately scored against historical data.

1. the phrase is broken up into ngram lengths
2. The array is reverse sorted so we compare 10 word length phrases first, then 9, and so on
3. Phrases are matched against positive, negative and neutral phrases in the relevant TXT files
4. Only matches that meet the minimum levenshtein\_min\_distance and similiarity\_min\_distance are kept