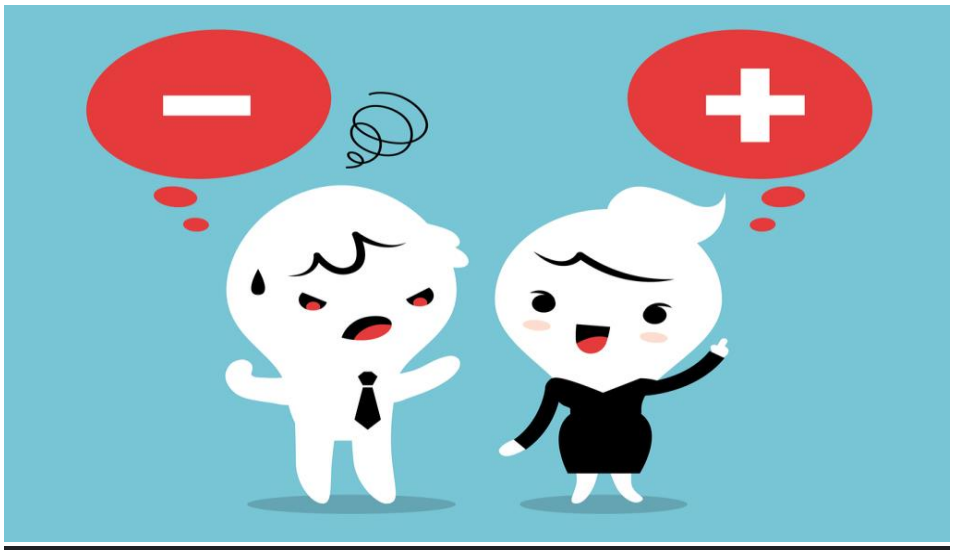
Sentiment Analysis

Project

For

Data Flair

By—Hemant Patar



What is sentiment analysis?

Sentiment Analysis is the process of determining whether a piece of writing is positive, negative or neutral by a computer. It's also known as **opinion mining.**

Why sentiment analysis is needed?

Sentiment Analysis has very wide application from (understanding a product's positive or negative impact on customer by their reviews) to (analysing the average positive or negative aspect of news or articles shared by any news website). Likewise there are many more applications to implement this. Here in this project I will be using this to find the sentiment of

a short story which can also be used on a large scale to categorise books, novels & articles in positive, negative and neutral segments.

What methods can be used for sentiment analysis?

From a technical point of view, we identified machine learning, lexicon-based, statistical and rule-based approaches.

• The machine learning method uses several learning algorithms to determine the sentiment by training on a known dataset.

• The lexicon-based approach involves calculating sentiment polarity for a review using the semantic orientation of words or sentences in the text. The "semantic orientation" is a measure of subjectivity and opinion in text.

• The rule-based approach looks for opinion words in a text and then classifies it based on the number of positive and negative words. It considers different rules for classification such as dictionary polarity, negation words, booster words, idioms, emoticons, mixed opinions

• Statistical models represent each review as a mixture of latent aspects and ratings. It is assumed that aspects and their ratings can be represented by multinomial distributions and try to cluster head terms into aspects and sentiments into ratings.

VADER Sentiment Analysis :

Here I have done Sentiment Analysis using VADER.

VADER (Valence Aware Dictionary and sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool. **VADER** uses a list of lexical features (e.g. word) which are labelled as positive or negative according to their semantic orientation to calculate the text sentiment. Vader sentiment returns the probability of a given input sentence to be positive, negative, and neutral.

Let's Start-

Here I am doing sentiment analysis of a short story -- The short story was taken from below website.
https://americanliterature.com/author/kate-chopin/short-story/the-night-came-slowly
The story is saved as text file for analysis named story.txt.

Before starting we must install all the libraries listed below and download the vader lexicon package as shown.

Use `nltk.download()` in Python with/without Jupyter:

```
>>> import nltk
>>> nltk.download('vader_lexicon')
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/liling.tan/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
True
```

Now we need to import below listed libraries in order to make it work.

```
import string
import nltk
from collections import Counter
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

import warnings
warnings.filterwarnings('ignore')
#nltk.download('vader_lexicon')
```

Now load the txt file containing the short story for analysis.

Pre-processing of the data is to be done first.

Convert all alphabets to lower case with ".lower()".

```
#Loading text file & converting it into lower case.
text = open('story.txt',encoding="utf-8").read()

lower_case = text.lower()
```

Here we remove all punctuations from the short story.

```
# str.maketrans removes any punctuations

cleaned_text = lower_case.translate(str.maketrans('', '', string.punctuation))
cleaned_text = cleaned_text.replace('“',' ').replace('”',' ').replace('’',' ').replace('–',' ')
```

The cleaned_text we got above can be used directly in *SentimentIntensityAnalyzer()* to get the sentiment of the text.

But to know the inside process of how Sentiment Intensity Analyzer works we see the below few steps.

Using word tokenize we tokenize sentence into words and remove stop words that is all unnecessary words as a part of cleaning the data for analysis.

After tokenizing and removing the stop words we get a bag of words in final_words from the cleaned text of the story.

```
# Using word_tokenize to tokenize sentence into words

tokenized_words = word_tokenize(cleaned_text, "english")
```

```
# Removing Stop Words
final_words = []

for word in tokenized_words:
    if word not in stopwords.words('english'):
        final_words.append(word)
```

Now we implement Lemmatization on the bag of words we gathered above.

Here we use Lemmatization to convert filtered out words to its base form - From plural to single + Base form of a word (example better-> good)

```
lemma_words = []

for word in final_words:
    word = WordNetLemmatizer().lemmatize(word)
    lemma_words.append(word)
```

We use another text file named emotion.txt with dictionary of some words with the emotion with respect to the word as value.

Eg. 'victimized': 'cheated', 'accused': 'cheated'.

Now we use our emotion.txt to seek matching words from the short story.

For every matching words we get its assigned emotion from the dictionary and its frequency of occurrence in the short story.

```
emotion_list = []

with open('emotions.txt', 'r') as file:
    for line in file:
        clear_line = line.replace("\n", '').replace(",", '').replace("'", '').strip()
        word, emotion = clear_line.split(':')

        if word in lemma_words:
            emotion_list.append(emotion)

print("People emotions from the text \n", emotion_list, '\n \n')


w = Counter(emotion_list)
print("Count of each emotion \n", w)
```

```
People emotions from the text
 [' fearless', ' hated']


Count of each emotion
 Counter({' fearless': 1, ' hated': 1})
```

Above we saw the basic process of how sentiment analyser works.

Now we use more sophisticated VADER to get the intensity of sentiment.

VADER's *SentimentIntensityAnalyzer()* takes in a string and returns a dictionary of scores in each of four categories:

- negative
- neutral
- positive
- compound *(computed by normalizing the scores above)*

Finally we use our sentiment intensity analyser to get the percentage of positive, negative or neutral inclination of the cleaned short story.

```
sia = SentimentIntensityAnalyzer()

sent = cleaned_text

print (sia.polarity_scores(sent))
```

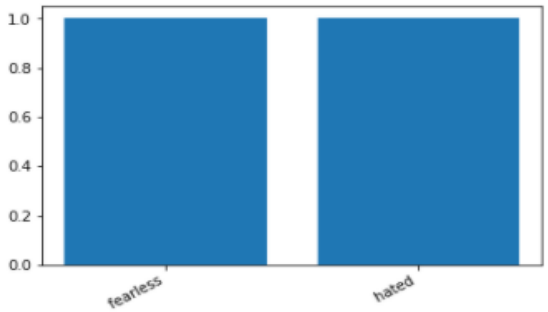
```
{'neg': 0.082, 'neu': 0.784, 'pos': 0.134, 'compound': 0.8993}
```

As a result we get a dictionary of negative, neutral, positive and compound percentages.

With the below set of code we can visualize the result obtained above in a more user (non-coder) friendly way instead of getting a dictionary.

```
def sentiment_analyse(sentiment_text):

    score = SentimentIntensityAnalyzer().polarity_scores(sentiment_text)

    if score['neg'] > score['pos']:
        print("\n      ******Negative Sentiment*******")

    elif score['neg'] < score['pos']:
        print("\n      ******Positive Sentiment*******")

    else:
        print("Neutral Sentiment")

sentiment_analyse(cleaned_text)

fig, ax1 = plt.subplots()
ax1.bar(w.keys(), w.values())
fig.autofmt_xdate()
# plt.savefig('graph.png')
plt.show()
```

```
      ******Positive Sentiment*******
```



Our short story is having a positive sentiment. Now we can know the sentiment of the story beforehand without reading it & one can decide whether to read it or not as per mood of individual.

Thank You.