Published in Towards Data Science

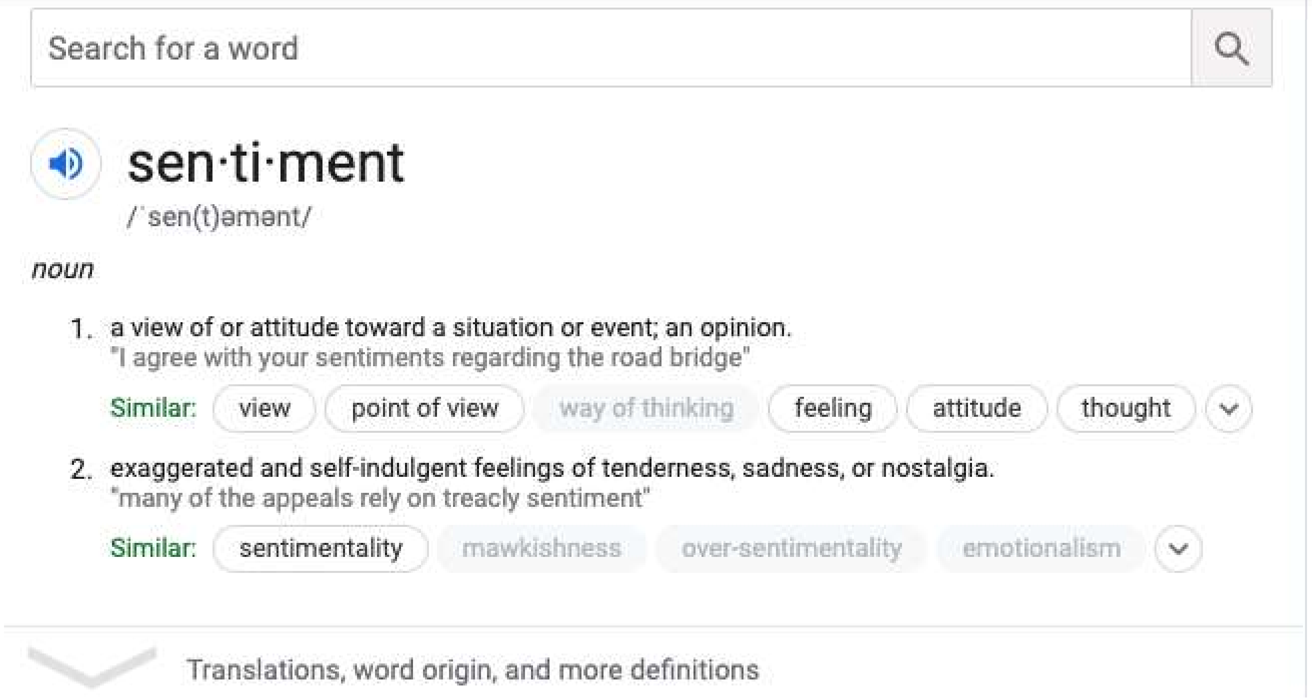
Parthvi Shah Follow
Jun 27, 2020 · 3 min read · Listen

Sentiment Analysis using TextBlob

Sentiment Analysis using TextBlob and its working!

source

What do you mean by Sentime



sen·ti·ment
/ˈsen(t)əmənt/
noun

1. a view of or attitude toward a situation or event; an opinion.
"I agree with your sentiments regarding the road bridge"
Similar: view, point of view, way of thinking, feeling, attitude, thought
2. exaggerated and self-indulgent feelings of tenderness, sadness, or nostalgia.
"many of the appeals rely on treacly sentiment"
Similar: sentimentality, mawkishness, over-sentimentality, emotionalism

Sentiment Analysis can help us decipher the mood and emotions of general public and gather insightful information regarding the context. Sentiment Analysis is a process of analyzing data and classifying it based on the need of the research.

These sentiments can be used for a better understanding of various events and impact caused by it. L. Bing [1] highlights that in the research literature it is possible to see many different names, e.g. "sentiment analysis, opinion mining, opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, review mining", however all of them have similar purposes and belong to the subject of sentiment analysis or opinion mining. By analysing these sentiments, we may find what people like, what they want and what their major concerns are.

Sentiment Analysis using TextBlob

TextBlob is a python library for Natural Language Processing (NLP).TextBlob actively used Natural Language ToolKit (NLTK) to achieve its tasks. NLTK is a library which gives an easy access to a lot of lexical resources and allows users to work with categorization, classification and many other tasks. TextBlob is a simple library which supports complex analysis and operations on textual data.

classifying negative and positive words. Generally, a text message will be represented by bag of words. After assigning individual scores to all the words, final sentiment is calculated by some pooling operation like taking an average of all the sentiments.

TextBlob returns **polarity** and **subjectivity** of a sentence. Polarity lies between [-1,1], -1 defines a negative sentiment and 1 defines a positive sentiment. Negation words reverse the polarity. TextBlob has semantic labels that help with fine-grained analysis. For example — emoticons, exclamation mark, emojis, etc. Subjectivity lies between [0,1]. **Subjectivity quantifies the amount of personal opinion and factual information contained in the text. The higher subjectivity means that the text contains personal opinion rather than factual information**. TextBlob has one more parameter — intensity. TextBlob calculates subjectivity by looking at the '**intensity**'. Intensity determines if a word *modifies* the next word. For English, adverbs are used as modifiers ('very good').

For example: We calculated polarity and subjectivity for "I do not like this example at all, it is too boring". For this particular example, polarity = -1 and subjectivity is 1, which is fair.

However, for the sentence "This was a helpful example but I would prefer another one". It returns **0.0 for both subjectivity and polarity** which is not the finest answer we'd expect.

It is expected that if the library returns exactly 0.0 either if your sentence didn't contain any words that had a polarity in the NLTK training set or because TextBlob uses a weighted average sentiment score over all the words in each sample. This easily diffuses out the effect of sentences with widely varying polarities between words in our case : 'helpful' and 'but'.

More often than not, for simple cases, TextBlob works just fine.

Steps to apply Sentiment Analysis using TextBlob –

2. Define a function that calculates subjectivity, polarity and give it a score based on the threshold you want to set.

```
def sentiment_analysis(tweet):
 def getSubjectivity(text):
   return TextBlob(text).sentiment.subjectivity

 #Create a function to get the polarity
 def getPolarity(text):
   return TextBlob(text).sentiment.polarity

 #Create two new columns ‘Subjectivity’ & ‘Polarity’
 tweet[‘TextBlob_Subjectivity’] =    tweet[‘tweet’].apply(getSubjectivity)
 tweet [‘TextBlob_Polarity’] = tweet[‘tweet’].apply(getPolarity)
 def getAnalysis(score):
  if score < 0:
    return ‘Negative’
  elif score == 0:
    return ‘Neutral’
  else:
    return ‘Positive’
 tweet [‘TextBlob_Analysis’] = tweet  [‘TextBlob_Polarity’].apply(getAnalysis )
return tweet
```

gistfile1.txt hosted with ❤ by GitHub view raw

Snippet of code for using TextBlob and analysis your data

Another useful article that I came across is: https://neptune.ai/blog/sentiment-analysis-python-textblob-vs-vader-vs-flair

References —