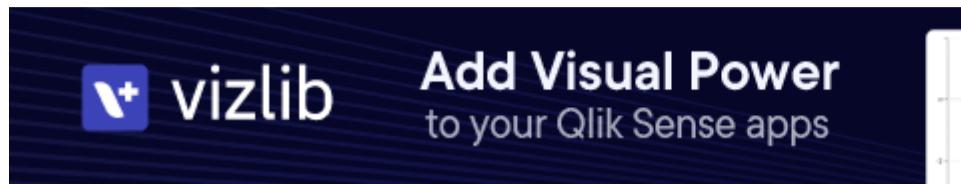


Sentiment Analysis Using Python



 Suvrat Arora – Updated On July 24th, 2023

[Beginner](#) [Machine Learning](#) [NLP](#) [Python](#)



Whether you speak of Twitter, Goodreads or Amazon—hardly is there a digital space not saturated with peoples' opinions. In today's world, it is crucial for organizations to dig into these opinions and get insights about their products or services. However, this data exists in such astounding amounts that gauging it manually is a next to impossible pursuit. This is where Data Science's yet another boon comes to play—**Sentiment Analysis**. In this article, we'll explore what sentiment analysis encompasses and the various ways to implement it in [Python](#).

This article was published as a part of the [Data Science Blogathon](#).



Table of contents

- [What is Sentiment Analysis?](#)
- [Gaining Insights and Making Decisions with Sentiment Analysis](#)
- [Sentiment Analysis Use Cases](#)
- [Ways to Perform Sentiment Analysis in Python](#)
- [Using Text Blob](#)
- [Using VADER](#)
- [Using Bag of Words Vectorization-Based Models](#)
- [Using LSTM-Based Models](#)
- [Using Transformer-Based Models](#)
- [Conclusion](#)
- [Frequently Asked Questions](#)

What is Sentiment Analysis?

Sentiment Analysis is a use case of [Natural Language Processing \(NLP\)](#) and comes under the category of [text classification](#). To put it simply, Sentiment Analysis involves classifying a text into various sentiments, such as positive or negative, Happy, Sad or Neutral, etc. Thus, the ultimate goal of sentiment analysis is to decipher the underlying mood, emotion, or sentiment of a text. This is also known as **Opinion Mining**.

Let us look at how a quick google search defines Sentiment Analysis:

Sentiment Analysis Using Python

noun

the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral.

"companies have key lessons to learn about harnessing the power of social media and sentiment analysis"

Feedback

See more →

How Does Sentiment Analysis Work?

Sentiment analysis typically works by employing natural language processing (NLP) techniques to analyze and understand the sentiment expressed in text. The process involves several steps:

1. **Text Preprocessing:** The text data is cleaned by removing irrelevant information, such as special characters, punctuation, and stopwords.
2. **Tokenization:** The text is divided into individual words or tokens to facilitate analysis.
3. **Feature Extraction:** Relevant features are extracted from the text, such as words, n-grams, or even parts of speech.
4. **Sentiment Classification:** Machine learning algorithms or pre-trained models are used to classify the sentiment of each text instance. This can be achieved through supervised learning, where models are trained on labeled data, or through pre-trained models that have learned sentiment patterns from large datasets.
5. **Post-processing:** The sentiment analysis results may undergo additional processing, such as aggregating sentiment scores or applying threshold rules to classify sentiments as positive, negative, or neutral.
6. **Evaluation:** The performance of the sentiment analysis model is assessed using evaluation metrics, such as accuracy, precision, recall, or F1 score.

Types of Sentiment Analysis

Various types of sentiment analysis can be performed, depending on the specific focus and objective of the analysis. Some common types include:



Ready to start your data science journey?
Master 23+ tools & learn with guided projects with the
Certified AI & ML BlackBelt Plus Program

[Enroll Now](#)

We use cookies on Analytics Vidhya websites to deliver our services, analyze web traffic, and improve your experience on the site. By using Analytics Vidhya, you

agree to our [Privacy Policy](#) and [Terms of Use](#).

[Accept](#)

Sentiment Analysis Using Python

document is analyzed. This type provides a more granular understanding of the sentiment expressed in different text parts.

3. Aspect-Based Sentiment Analysis: This approach focuses on identifying and extracting the sentiment associated with specific aspects or entities mentioned in the text. For example, in a product review, the sentiment towards different features of the product (e.g., performance, design, usability) can be analyzed separately.

4. Entity-Level Sentiment Analysis: This type of analysis identifies the sentiment expressed towards specific entities or targets mentioned in the text, such as people, companies, or products. It helps understand the sentiment associated with different entities within the same document.

5. Comparative Sentiment Analysis: This approach involves comparing the sentiment between different entities or aspects mentioned in the text. It aims to identify the relative sentiment or preferences expressed towards various entities or features.

Gaining Insights and Making Decisions with Sentiment Analysis

Well, by now I guess we are somewhat accustomed to what sentiment analysis is. But what is its significance and how do organizations benefit from it? Let us try and explore the same with an example. Assume you start a company that sells perfumes on an online platform. You put up a wide range of fragrances out there and soon customers start flooding in. After some time you decide to change the pricing strategy of perfumes—you plan to increase the prices of the popular fragrances and at the same time offer discounts on unpopular ones. Now, in order to determine which fragrances are popular, you start going through customer reviews of all the fragrances. But you're stuck! They are just so many that you cannot go through them all in one lifetime. This is where sentiment analysis can rope you out of the pit.

You simply gather all the reviews in one place and apply sentiment analysis to it. The following is a schematic representation of sentiment analysis on the reviews of three fragrances of perfumes—Lavender, Rose, and Lemon. (Please note that these reviews might have incorrect spellings, grammar, and punctuations as it is in the real-world scenarios)



We use cookies on Analytics Vidhya websites to deliver our services, analyze web traffic, and improve your experience on the site. By using Analytics Vidhya, you agree to our [Privacy Policy](#) and [Terms of Use](#).

[Accept](#)

Sentiment Analysis Using Python

means your company **should not change its pricing**.

- **Fragrance-3 (Lemon)** has an overall **negative** sentiment associated with it—thus, your company should **consider offering a discount** on it to balance the scales.

This was just a simple example of how sentiment analysis can help you gain insights into your products/services and help your organization make decisions.

Sentiment Analysis Use Cases

We just saw how sentiment analysis can empower organizations with insights that can help them make data-driven decisions. Now, let's peep into some more use cases of sentiment analysis:

1. **Social Media Monitoring for Brand Management:** Brands can use sentiment analysis to gauge their Brand's public outlook. For example, a company can gather all Tweets with the company's mention or tag and perform sentiment analysis to learn the company's public outlook.
2. **Product/Service Analysis:** Brands/Organizations can perform sentiment analysis on customer reviews to see how well a product or service is doing in the market and make future decisions accordingly.
3. **Stock Price Prediction:** Predicting whether the stocks of a company will go up or down is crucial for investors. One can determine the same by performing sentiment analysis on News Headlines of articles containing the company's name. If the news headlines pertaining to a particular organization happen to have a positive sentiment—its stock prices should go up and vice-versa.

Ways to Perform Sentiment Analysis in Python

Python is one of the most powerful tools when it comes to performing data science tasks—it offers a multitude of ways to perform sentiment analysis. The most popular ones are enlisted here:

1. Using Text Blob
2. Using Vader
3. Using Bag of Words Vectorization-based Models
4. Using LSTM-based Models
5. Using Transformer-based Models

Let's dive deep into them one by one.

Note: For the purpose of demonstrations of methods 3 & 4 (Using Bag of Words Vectorization-based Models and Using LSTM-based Models) sentiment analysis has been used. It comprises more than 5000 text excerpts labelled as positive, negative or neutral. The

Sentiment Analysis Using Python

Polarity determines the sentiment of the text. Its values lie in [-1,1] where -1 denotes a highly negative sentiment and 1 denotes a highly positive sentiment.

Subjectivity determines whether a text input is factual information or a personal opinion. Its value lies between [0,1] where a value closer to 0 denotes a piece of factual information and a value closer to 1 denotes a personal opinion.

Installation:

```
pip install textblob
```

Importing Text Blob:

```
from textblob import TextBlob
```

Code Implementation for Sentiment Analysis Using Text Blob:

Writing code for sentiment analysis using TextBlob is fairly simple. Just import the TextBlob object and pass the text to be analyzed with appropriate attributes as follows:

Python Code:

```
main.py
from sklearn import svm
import pandas as pd
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')

# Read Train and Test datasets
# Login/ Signup to View & Run Code in the browser
# View & Run Code
```

Subjectivity of Text 2 is 1.0

Using VADER

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a rule-based sentiment analyzer that has been trained on social media text. Just like Text Blob, its usage in Python is pretty simple. We'll see its usage in code implementation with an example in a while.

Installation:

```
pip install vaderSentiment
```

Importing SentimentIntensityAnalyzer class from Vader:

We use cookies on Analytics Vidhya websites to deliver our services, analyze web traffic, and improve your experience on the site. By using Analytics Vidhya, you

agree to our [Privacy Policy](#) and [Terms of Use](#).

Sentiment Analysis Using Python

SentimentIntensityAnalyzer

Code for Sentiment Analysis Using Vader:

Firstly, we need to create an object of the SentimentIntensityAnalyzer class; then we need to pass the text to the polarity_scores() function of the object as follows:

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
sentiment = SentimentIntensityAnalyzer()
text_1 = "The book was a perfect balance between writing style and plot."
text_2 = "The pizza tastes terrible."
sent_1 = sentiment.polarity_scores(text_1)
sent_2 = sentiment.polarity_scores(text_2)
print("Sentiment of text 1:", sent_1)
print("Sentiment of text 2:", sent_2)
```

Output:

```
Sentiment of text 1: {'neg': 0.0, 'neu': 0.73, 'pos': 0.27, 'compound': 0.5719}
Sentiment of text 2: {'neg': 0.508, 'neu': 0.492, 'pos': 0.0, 'compound': -0.4767}
```

As we can see, a VaderSentiment object returns a dictionary of sentiment scores for the text to be analyzed.

Using Bag of Words Vectorization-Based Models

In the two approaches discussed as yet i.e. Text Blob and Vader, we have simply used Python libraries to perform sentiment analysis. Now we'll discuss an approach wherein we'll train our own model for the task. The steps involved in performing sentiment analysis using the Bag of Words Vectorization method are as follows:

1. Pre-Process the text of training data (Text pre-processing involves Normalization, Tokenization, Stopwords Removal, and Stemming/Lemmatization.)
2. Create a Bag of Words for the pre-processed text data using the Count Vectorization or TF-IDF Vectorization approach.
3. Train a suitable classification model on the processed data for sentiment classification.

Code for Sentiment Analysis using Bag of Words Vectorization Approach:

To build a sentiment analysis model using the BOW Vectorization Approach we need a labeled dataset. As stated earlier, the dataset used for this demonstration has been obtained from Kaggle. We have simply used sklearn's count vectorizer to create the BOW. After, we trained a Multinomial Naive Bayes classifier, for which an accuracy score of 0.84 was obtained.

Dataset can be obtained from [here](#).

Sentiment Analysis Using Python

```

token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer(stop_words='english', ngram_range = (1,1), tokenizer =
token.tokenize)
text_counts = cv.fit_transform(data['sentences'])
#Splitting the data into training and testing
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(text_counts, data['feedback'],
test_size=0.25, random_state=5)
#Training the model
from sklearn.naive_bayes import MultinomialNB
MNB = MultinomialNB()
MNB.fit(X_train, Y_train)
#Calculating the accuracy score of the model
from sklearn import metrics
predicted = MNB.predict(X_test)
accuracy_score = metrics.accuracy_score(predicted, Y_test)
print("Accuracy Score: ",accuracy_score)

```

Output:

Accuracy Score: 0.9111675126903553

The trained classifier can be used to predict the sentiment of any given text input.

Using LSTM-Based Models

Though we were able to obtain a decent accuracy score with the Bag of Words Vectorization method, it might fail to yield the same results when dealing with larger datasets. This gives rise to the need to employ deep learning-based models for the training of the sentiment analysis model.

For NLP tasks we generally use RNN-based models since they are designed to deal with sequential data. Here, we'll train an LSTM (Long Short Term Memory) model using **TensorFlow with Keras**. The steps to perform sentiment analysis using LSTM-based models are as follows:

1. Pre-Process the text of training data (Text pre-processing involves Normalization, Tokenization, Stopwords Removal, and Stemming/Lemmatization.)
2. Import **Tokenizer** from `Keras.preprocessing.text` and create its object. Fit the tokenizer on the entire training text (so that the Tokenizer gets trained on the training data vocabulary). Generated text embeddings using the `texts_to_sequences()` method of the Tokenizer and store them after padding them to an equal length. (Embeddings are numerical/vectorized representations of text. Since we cannot feed our model with the text data directly, we first need to convert them to embeddings)
3. After having generated the embeddings we are ready to build the model. We build the model using TensorFlow—add Input, LSTM, and dense layers to it. Add dropouts and tune the hyperparameters to get a decent accuracy score. Generally, we tend to use

Sentiment Analysis Using Python

Model

Here, we have used the same dataset as we used in the case of the BOW approach. A training accuracy of 0.90 was obtained.

```
#Importing necessary libraries
import nltk
import pandas as pd
from textblob import Word
from nltk.corpus import stopwords
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
from keras.models import Sequential
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from sklearn.model_selection import train_test_split
#Loading the dataset
data = pd.read_csv('Finance_data.csv')
#Pre-Processing the text
def cleaning(df, stop_words):
    df['sentences'] = df['sentences'].apply(lambda x: ' '.join(x.lower() for x in
x.split()))
    # Replacing the digits/numbers
    df['sentences'] = df['sentences'].str.replace('d', '')
    # Removing stop words
    df['sentences'] = df['sentences'].apply(lambda x: ' '.join(x for x in x.split()
if x not in stop_words))
    # Lemmatization
    df['sentences'] = df['sentences'].apply(lambda x: ' '.join([Word(x).lemmatize()
for x in x.split()]))
    return df
stop_words = stopwords.words('english')
data_cleaned = cleaning(data, stop_words)
#Generating Embeddings using tokenizer
tokenizer = Tokenizer(num_words=500, split=' ')
tokenizer.fit_on_texts(data_cleaned['verified_reviews'].values)
X = tokenizer.texts_to_sequences(data_cleaned['verified_reviews'].values)
X = pad_sequences(X)
#Model Building
model = Sequential()
model.add(Embedding(500, 120, input_length = X.shape[1]))
model.add(SpatialDropout1D(0.4))
model.add(LSTM(704, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(352, activation='LeakyReLU'))
model.add(Dense(3, activation='softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics =
['accuracy'])
print(model.summary())
#Model Training
model.fit(X_train, y_train, epochs = 20, batch_size=32, verbose =1)
#Model Testing
model.evaluate(X_test,y_test)
```

[Top 10 blogs on NLP in Analytics](#)
[Vidhya 2022](#)

[Sentiment Analysis: VADER or TextBlob?](#)

[Using Data Science to Identify Top Twitter Influencers](#)

[Sentiment Analysis with NLP](#)

Sentiment Analysis Using Python

model from scratch, it is quite tedious a task. Thus, we can use pre-trained transformer models available on [Hugging Face](#). Hugging Face is an open-source AI community that offers a multitude of pre-trained models for NLP applications. These models can be used as such or can be fine-tuned for specific tasks.

Installation:

```
pip install transformers
```

Importing SentimentIntensityAnalyzer class from Vader:

```
import transformers
```

Code for Sentiment Analysis Using Transformer based models:

To perform any task using transformers, we first need to import the pipeline function from transformers. Then, an object of the pipeline function is created and the task to be performed is passed as an argument (i.e sentiment analysis in our case). We can also specify the model that we need to use to perform the task. Here, since we have not mentioned the model to be used, the distillery-base-uncased-finetuned-sst-2-English mode is used by default for sentiment analysis. You can check out the list of available tasks and models [here](#).

```
from transformers import pipeline
sentiment_pipeline = pipeline("sentiment-analysis")
data = ["It was the best of times.", "t was the worst of times."]
sentiment_pipeline(data)
```

Output:

```
[{'label': 'POSITIVE', 'score': 0.999457061290741}, {'label': 'NEGATIVE', 'score': 0.9987301230430603}]
```

Conclusion

In this age when users can express their viewpoints effortlessly and data is generated in superfluity in just fractions of seconds—drawing insights from such data is vital for organizations to make efficient decisions—and Sentiment Analysis proves to be the missing piece of the puzzle!

By now we have covered in great detail what exactly sentiment analysis entails and the various methods one can use to perform it in Python. But these were just some rudimentary demonstrations—you must surely go ahead and fiddle with the models and try them out on your own data.

Frequently Asked Questions

Q1. What do you mean by sentiment analysis?

A. Sentiment analysis means extracting and determining a text's sentiment or emotional

Sentiment Analysis Using Python

A. The two types of sentiment analysis are (1) Document-level sentiment analysis, which analyzes the sentiment of an entire document, and (2) Sentence-level sentiment analysis, which focuses on analyzing the sentiment of individual sentences within a document.

Q4. What is sentiment analysis and its types?

A. Sentiment analysis is analyzing and classifying the sentiment expressed in text. It can be categorized into document-level and sentence-level sentiment analysis, where the former analyzes the sentiment of a whole document, and the latter focuses on the sentiment of individual sentences.

The media shown in this article is not owned by Analytics Vidhya and is used at the Author's discretion.

[blogathon](#) [sentiment analysis](#) [Sentiment Analysis importance](#)

About the Author



[Suvrat Arora](#)

Our Top Authors



Download

Analytics Vidhya App for the Latest blog/Article



Previous Post

Next Post

[Why is Face Alignment Important for Face Recognition?](#)

[Building Smarter Solutions with Machine Learning](#)

Sentiment Analysis Using Python

Notify me of follow-up comments by email.

Notify me of new posts by email.

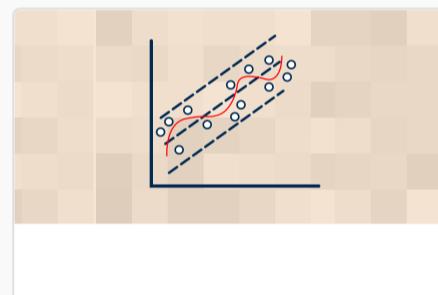
Submit

Top Resources



[10 Best AI Image Generator Tools to Use in 2023](#)

avcontentteam -
AUG 17, 2023



[Everything you need to Know about Linear Regression!](#)

KAVITA MALI -
OCT 04, 2021



[How to Read and Write With CSV Files in Python?](#)

Harika Bonthu -
AUG 21, 2021



[Skewness and Kurtosis: Quick Guide \(Updated 2023\)](#)

Suvarna Gawali -
MAY 02, 2021



Download App



Analytics Vidhya

[About Us](#)

[Our Team](#)

[Careers](#)

[Contact us](#)

Data Scientists

[Blog](#)

[Hackathon](#)

[Join the Community](#)

[Apply Jobs](#)

Companies

[Post Jobs](#)

[Trainings](#)

[Hiring Hackathons](#)

[Advertising](#)

Visit us

