Twitter Sentiment Analysis Using Natural Language Processing (NLP) with Python

The Ultimate Guide for Identifying Sentiments and Performing Text Analysis on Twitter Data

Blog By: Chaithanya Vamshi Sai

Student ID: 21152797

1. Introduction

Natural Language Processing (NLP) is an emerging field and a subset of machine learning which aims to train computers to understand human languages. The most common application of NLP is Sentiment Analysis.

In the process of NLP, we aim to prepare a textual dataset to build a vocabulary for text classification.

In this blog, I will walk you through the entire process of how to do Twitter Sentiment Analysis using Python.

2. What is Sentiment Analysis?

Sentiment Analysis is also known as opinion mining which is one of the applications of NLP. It is a set of methods and techniques used to extract information from text or speech. In simpler terms, it involves classifying a piece of text as positive, negative or neutral.

Twitter is one of those social media platforms where people are free to share their opinions. We mostly see negative opinions on Twitter. So, we should continue to analyse the sentiments to find the type of people who are spreading hate and negativity.

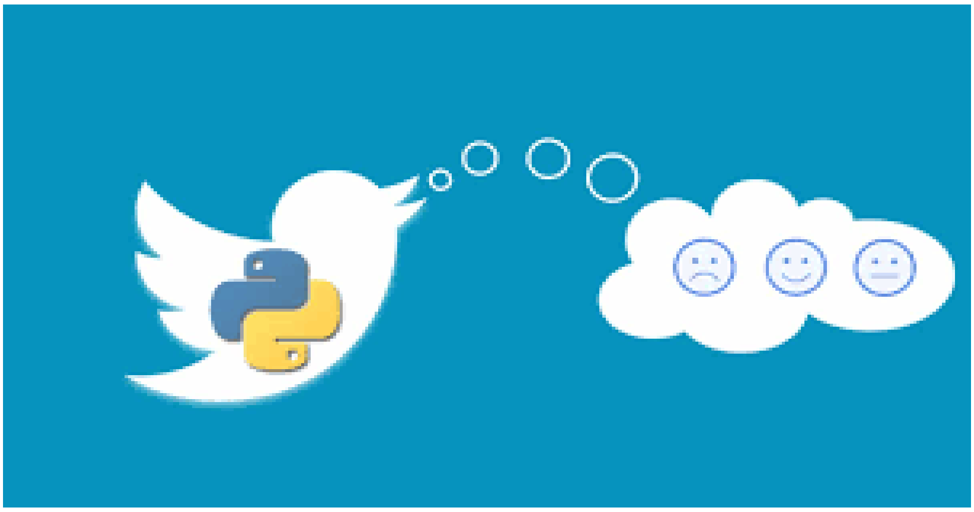


Image Source: https://giters.com/

3. Problem Statement

The objective of this task is we are given a data set of tweets that are labelled as positive and negative. Using these labelled data, we need to train a Machine learning model using Python to predict the sentiment (positive or negative) of new tweets.

4. Importing Libraries and Reading Data

```
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

%matplotlib inline
```

```
train  = pd.read_csv('/content/train.csv')
test = pd.read_csv('/content/test.csv')
```

5. Tweets Pre-processing and Cleaning

The pre-processing of the text data is an essential step as it makes it easier to extract information from the text and apply machine learning algorithms to it.

The objective of this step is to Inspect data and clean noise that is irrelevant to find the sentiment of tweets such as punctuation, special characters, numbers, and terms that don't carry much weightage in context to the text.

Data Pre-processing is divided into two parts:

1. Data Inspection
2. Data Cleaning

1. Data Inspection

- Checking the first few rows of the training dataset.

```
train.head()
```

	Id	Text	Sentiment
0	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative
1	088c60f138	my boss is bullying me...	negative
2	9642c003ef	what interview! leave me alone	negative
3	358bd9e861	Sons of ****, why couldn`t they put them on t...	negative
4	6e0c6d75b1	2am feedings for the baby are fun when he is a...	positive

- Checking the first 5 rows of the Negative tweets in the training dataset

```
train[train['Sentiment']== 'negative'].head(5)
```

	Id	Text	Sentiment
0	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative
1	088c60f138	my boss is bullying me...	negative
2	9642c003ef	what interview! leave me alone	negative
3	358bd9e861	Sons of ****, why couldn`t they put them on t...	negative
7	74a76f6e0a	My Sharpie is running DANGERously low on ink	negative

- Checking the first 5 rows of the Positive tweets in the training dataset

```
train[train['Sentiment']== 'positive'].head(5)
```

	Id	Text	Sentiment
4	6e0c6d75b1	2am feedings for the baby are fun when he is a...	positive
5	fc2cbefa9d	Journey!? Wow... u just became cooler. hehe....	positive
6	16fab9f95b	I really really like the song Love Story by Ta...	positive
13	e48b0b8a23	Playing Ghost Online is really interesting. Th...	positive
14	e00c6ef376	the free fillin` app on my ipod is fun, im add...	positive

- Checking the Shape and Distribution of Tweets in the training dataset

```
train.shape, test.shape
```

```
((16363, 3), (1000, 2))
```

```
train['Sentiment'].value_counts()
```

```
positive    8582
negative    7781
Name: Sentiment, dtype: int64
```

2. Data Cleaning

For our convenience, let's first combine train and test sets. This saves the trouble of performing the same steps twice on test and train.

```
combi = train.append(test,ignore_index = True)
```

```
combi.shape
```

```
(17363, 3)
```

a) Removing Twitter Handles (@user)

```
def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for i in r:
        input_txt = re.sub(i, '', input_txt)

    return input_txt
```

```
# remove twitter handles (@user)
combi['tidy_tweet'] = np.vectorize(remove_pattern)(combi['Text'], "@[\w]*")
```

b) Removing Punctuations, Numbers, and Special Characters

```
# remove special characters, numbers, punctuations
combi['tidy_tweet'] = combi['tidy_tweet'].str.replace("[^a-zA-Z#]", " ")
```

c) Removing Short Words

```
combi['tidy_tweet'] = combi['tidy_tweet'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))
```

```
combi.head()
```

	Id	Text	Sentiment	tidy_tweet
0	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative	Sooo will miss here Diego
1	088c60f138	my boss is bullying me...	negative	boss bullying
2	9642c003ef	what interview! leave me alone	negative	what interview leave alone
3	358bd9e861	Sons of ****, why couldn`t they put them on t...	negative	Sons couldn they them releases already bought
4	6e0c6d75b1	2am feedings for the baby are fun when he is a...	positive	feedings baby when smiles coos

We can see the difference between the actual tweets and the cleaned tweets (tidy_tweet) quite clearly. Only the important words in the tweets have been retained and the noise (numbers, punctuations, and special characters) has been removed.

d) Tokenization

Now we will tokenize all the cleaned tweets in our dataset. Tokens are individual terms or words, and tokenization is the process of splitting a string of text into tokens.

```
tokenized_tweet = combi['tidy_tweet'].apply(lambda x: x.split())
tokenized_tweet.head()
```

--NORMAL--

```
0                    [Sooo, will, miss, here, Diego]
1                                    [boss, bullying]
2                    [what, interview, leave, alone]
3    [Sons, couldn, they, them, releases, already, ...
4                 [feedings, baby, when, smiles, coos]
Name: tidy_tweet, dtype: object
```

e) Text Normalization (Stemming)

Stemming is a rule-based process of stripping the suffixes ("ing", "ly", "es", "s" etc) from a word.

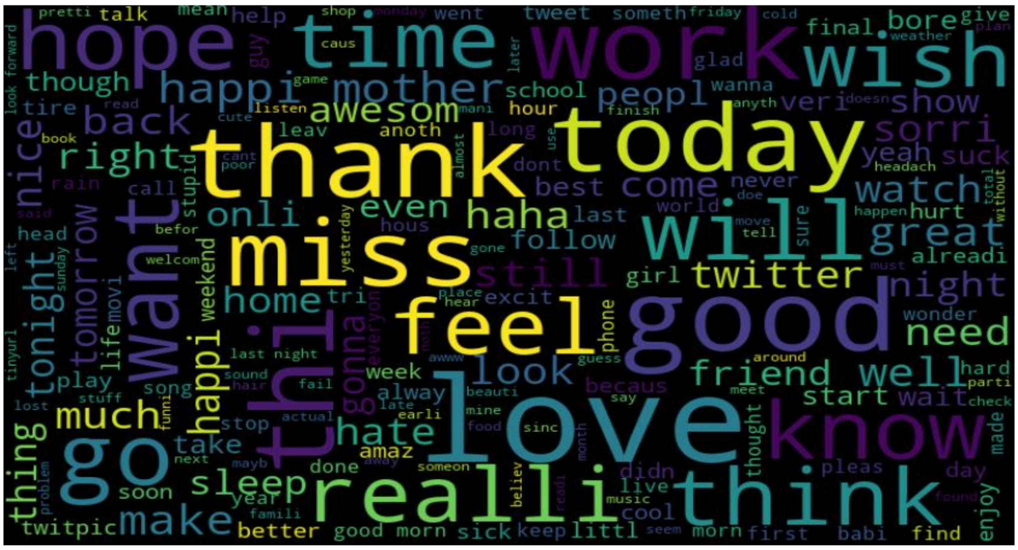
```
from nltk.stem.porter import *
stemmer = PorterStemmer()

tokenized_tweet = tokenized_tweet.apply(lambda x: [stemmer.stem(i) for i in x]) # stemming
tokenized_tweet.head()
```

```
0                       [sooo, will, miss, here, diego]
1                                          [boss, bulli]
2                          [what, interview, leav, alon]
3    [son, couldn, they, them, releas, alreadi, bou...
4                        [feed, babi, when, smile, coo]
Name: tidy_tweet, dtype: object
```

6. Data Exploration and Visualization from Tweets

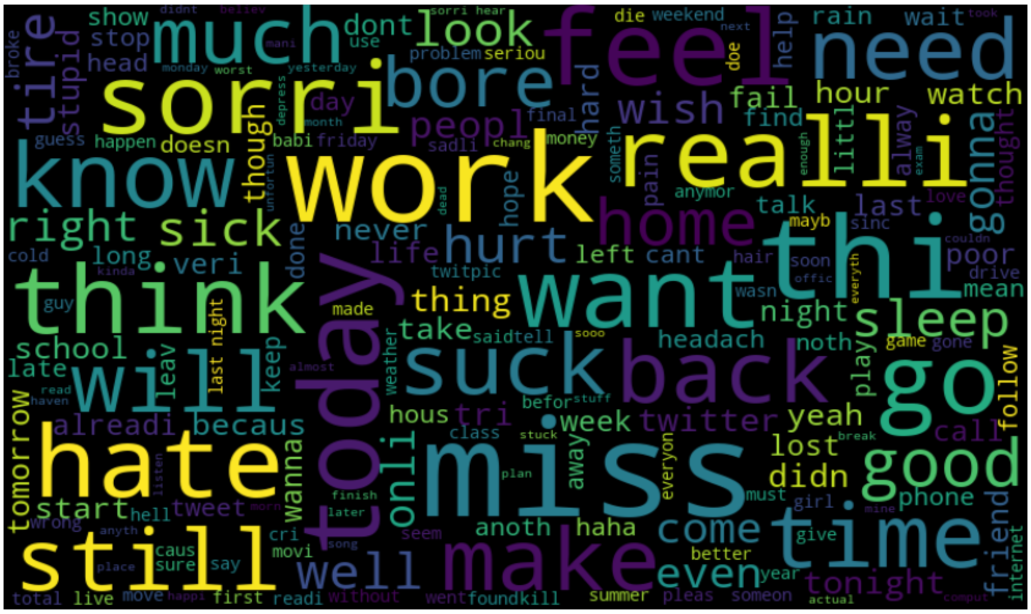
Exploring and visualizing data, no matter whether it's text or any other data is an essential step in gaining insights. we must think and ask questions related to the data in hand.

1. The common words used in the tweets: Word Cloud



2. Words in Positive Tweets: Word Cloud

3. Words in Negative Tweets: Word Cloud



7. Extracting Features from Cleaned Tweets

To analyse pre-processed data, it needs to be converted into features. Depending upon the usage, text features can be constructed using assorted. In this blog, we will be covering Bag-of-Words and TF-IDF.

1. Bag-of-Words Features

```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
import gensim
```

```
]   bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='english')
    bow = bow_vectorizer.fit_transform(combi['tidy_tweet'])
    bow.shape
```

```
(17363, 1000)
```

2. TF-IDF Features

```
tfidf_vectorizer = TfidfVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='english')
tfidf = tfidf_vectorizer.fit_transform(combi['tidy_tweet'])
tfidf.shape
```

```
(17363, 1000)
```

8. Model Building: Twitter Sentiment Analysis

Now we will build predictive models on the dataset using the two-feature set — Bag-of-Words and TF-IDF.

1. Model Building using Bag-of-Words Features

a) Logistic Regression
b) Decision Tree Classifier
c) Random Forest Classifier
d) SVM Classifier
e) XGBoost Classifier

2. Model Building using TF-IDF Features

a) Logistic Regression
b) Decision Tree Classifier
c) Random Forest Classifier
d) SVM Classifier
e) XGBoost Classifier

3. Summary of Accuracy Scores on Training Data Set

Below is the summary table showing accuracy scores on Training Data set for different Machine learning models and feature extraction techniques.

From all the models, the SVM classifier with TFI-IDF features achieved the best Accuracy of **94.648%** on Training data and 84.029% accuracy on validation data.

Feature Extraction	Model	Training Accuracy	Validation Accuracy
Bag of Words	Logistic Regression	87.515	83.662
	Decision Tree Classifier	98.044	79.364
	Random Forest Classifier	98.044	82.134
	SVM Classifier	93.059	83.581
	XGBoost Classifier	79.841	79.079
TF-IDF	Logistic Regression	87.401	83.907
	Decision Tree Classifier	98.018	79.119
	Random Forest Classifier	98.018	82.74
	SVM Classifier	94.648	84.029
	XGBoost Classifier	80.094	79.079

9. Predictions on Test Data set

Using the best model SVM classifier, we will make predictions on the test data set and save the predictions to a .csv file named test-predictions.csv.

```
predictions = svc_tf.predict(test_tfidf)

test['Sentiment'] = predictions

test.to_csv('/content/test-predictions.csv', index = False)

df1 = pd.read_csv('/content/test-predictions.csv')
df1
```

Test Data Predictions Data frame saved in a .csv file with columns Id, Text & Sentiment.

	Id	Text	Sentiment
0	96d74cb729	Shanghai is also really exciting (precisely -...	positive
1	eee518ae67	Recession hit Veronique Branquinho, she has to...	negative
2	01082688c6	happy bday!	positive
3	33987a8ee5	http://twitpic.com/4w75p - I like it!!	positive
4	726e501993	that`s great!! weee!! visitors!	positive
...	...	...	...
995	9b210c4a6f	Haha...YAY!!! I`M CURED!!!!	positive
996	68c674acdb	Sick, sick, sick. This sucks. i can`t even bre...	negative
997	6cadda7b98	Adding names to my Twitter account and learnin...	positive
998	79a28b1ac7	ooh thats an early start ive got bed planned...	negative
999	05a1b09ce9	Booo my best friend is leavin for the weekend....	positive

1000 rows × 3 columns

10. Conclusion

In this blog, we have learnt how to approach a Sentiment Analysis problem. We started with pre-processing and exploration of data. Then we extracted features from the cleaned text using Bag-of-Words and TF-IDF.

Finally, we were able to apply different Machine Learning models using both the feature sets to classify the tweets and make predictions on the test data set using the best Machine learning model which outperformed the other models.

Source Code of the Project Link: **GitHub**

11. References

https://www.analyticsvidhya.com/blog/2021/07/understanding-natural-language-processing-a-beginners-guide/

https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/

https://thecleverprogrammer.com/2021/09/13/twitter-sentiment-analysis-using-python/