Manoj Kumar

Sentiment Analysis using Python

for Beginners

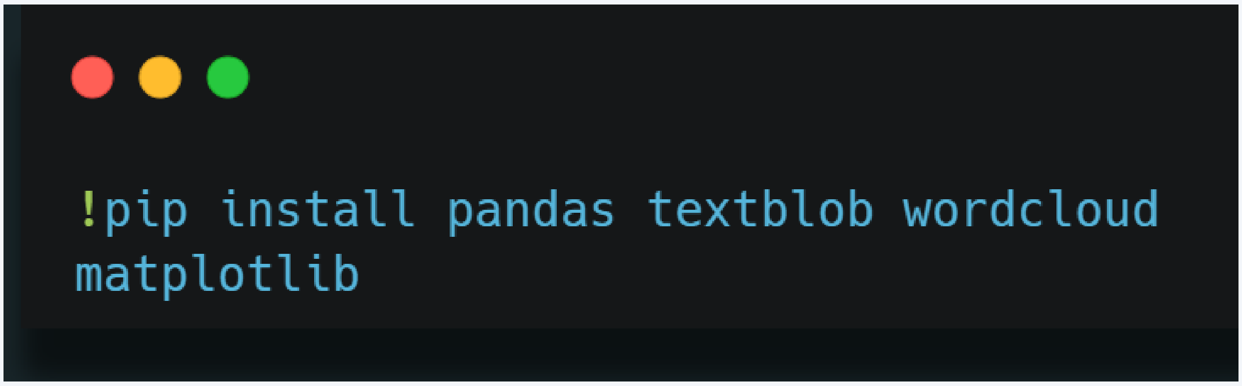
SWIPE

In this guide, we will walk through the process of performing sentiment analysis and generating a word cloud using comments from my LinkedIn post.

SWIPE

Prerequisites

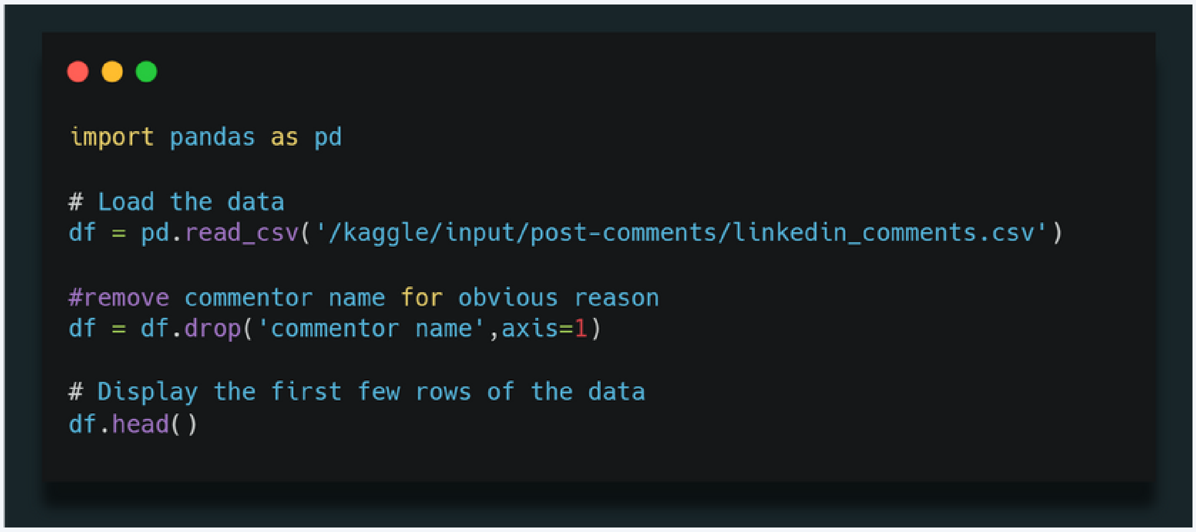
- Python installed on your system.
- Jupyter Notebook or any Python IDE for executing the code.
- The comments from your LinkedIn post saved as a CSV.



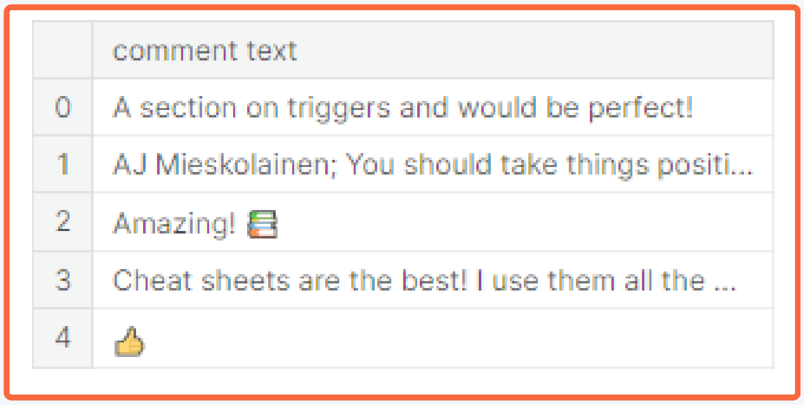
```
!pip install pandas textblob wordcloud
matplotlib
```

SWIPE

Load the Comments Data



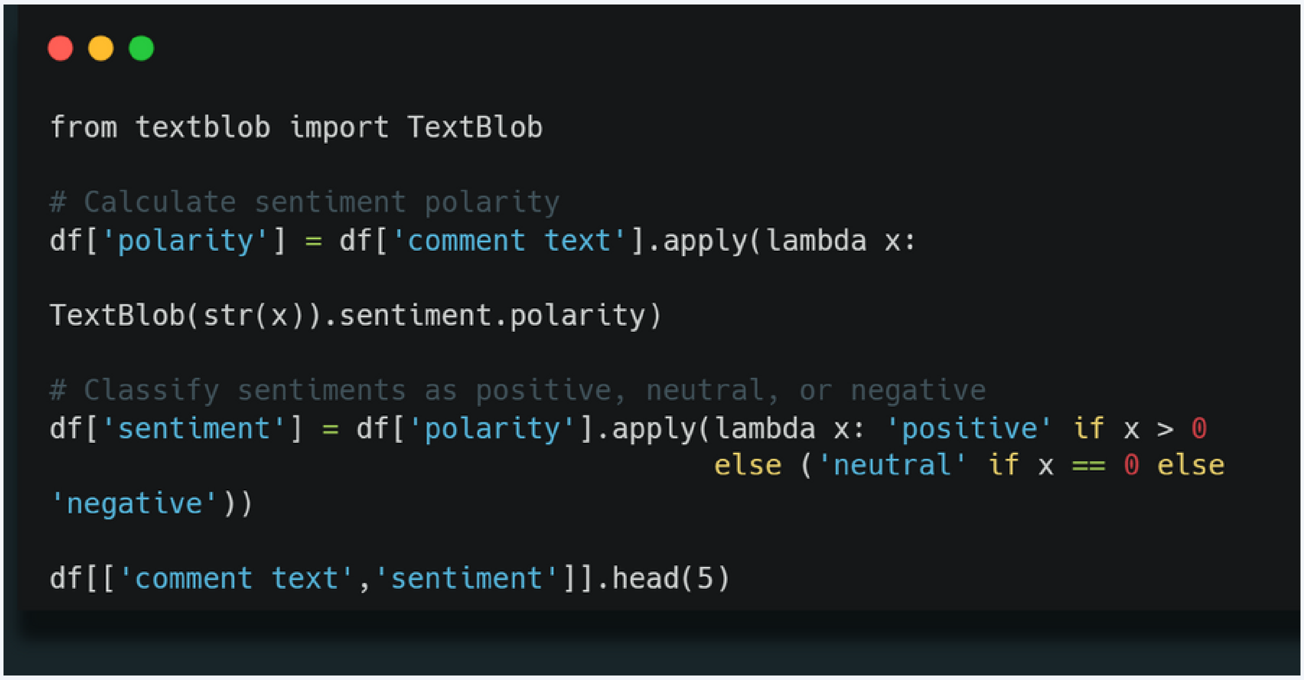
```python
import pandas as pd

# Load the data
df = pd.read_csv('/kaggle/input/post-comments/linkedin_comments.csv')

#remove commentor name for obvious reason
df = df.drop('commentor name',axis=1)

# Display the first few rows of the data
df.head()
```



	comment text
0	A section on triggers and would be perfect!
1	AJ Mieskolainen; You should take things positi...
2	Amazing! 📚
3	Cheat sheets are the best! I use them all the ...
4	👍

SWIPE

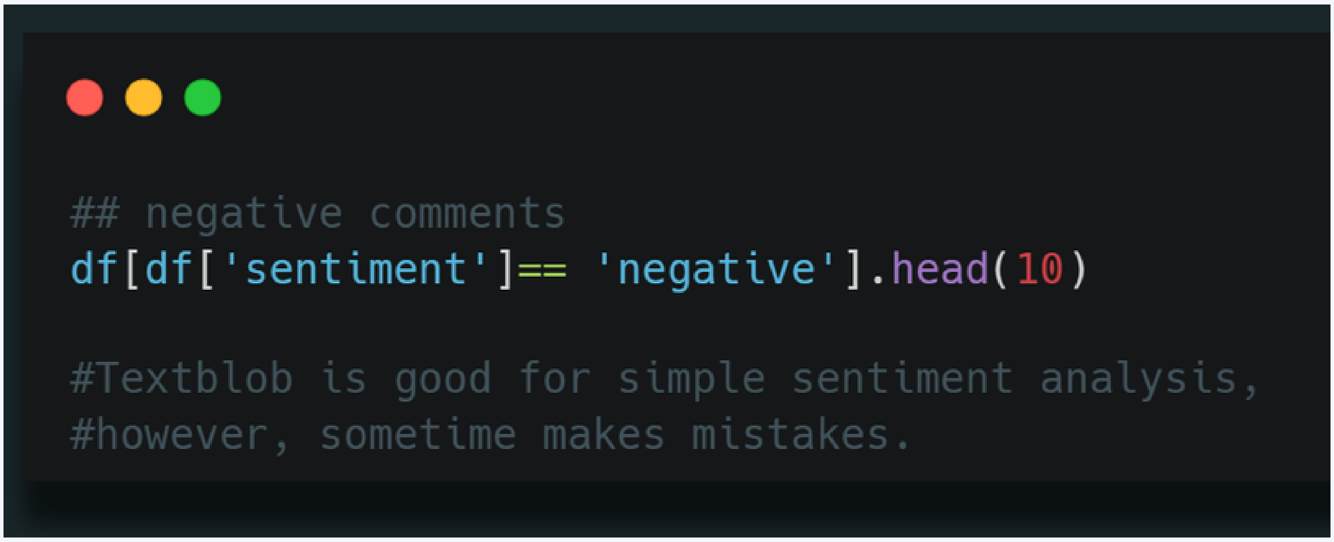
Perform Sentiment Analysis



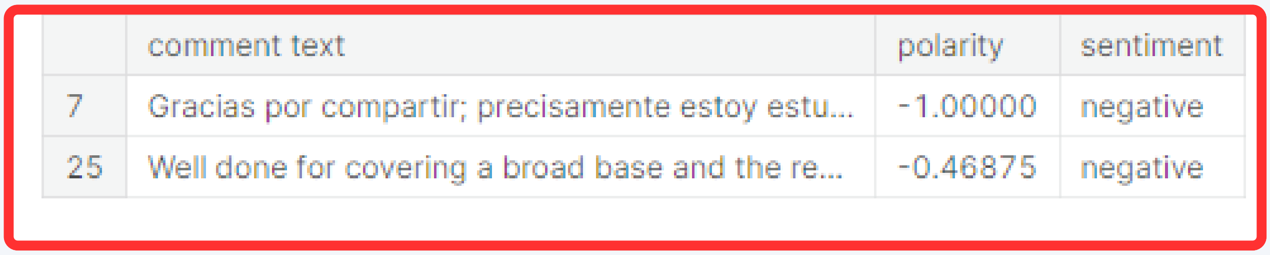
```
from textblob import TextBlob

# Calculate sentiment polarity
df['polarity'] = df['comment text'].apply(lambda x:

TextBlob(str(x)).sentiment.polarity)

# Classify sentiments as positive, neutral, or negative
df['sentiment'] = df['polarity'].apply(lambda x: 'positive' if x > 0
                                       else ('neutral' if x == 0 else
'negative'))

df[['comment text','sentiment']].head(5)
```

SWIPE

Let’s see negative comments.



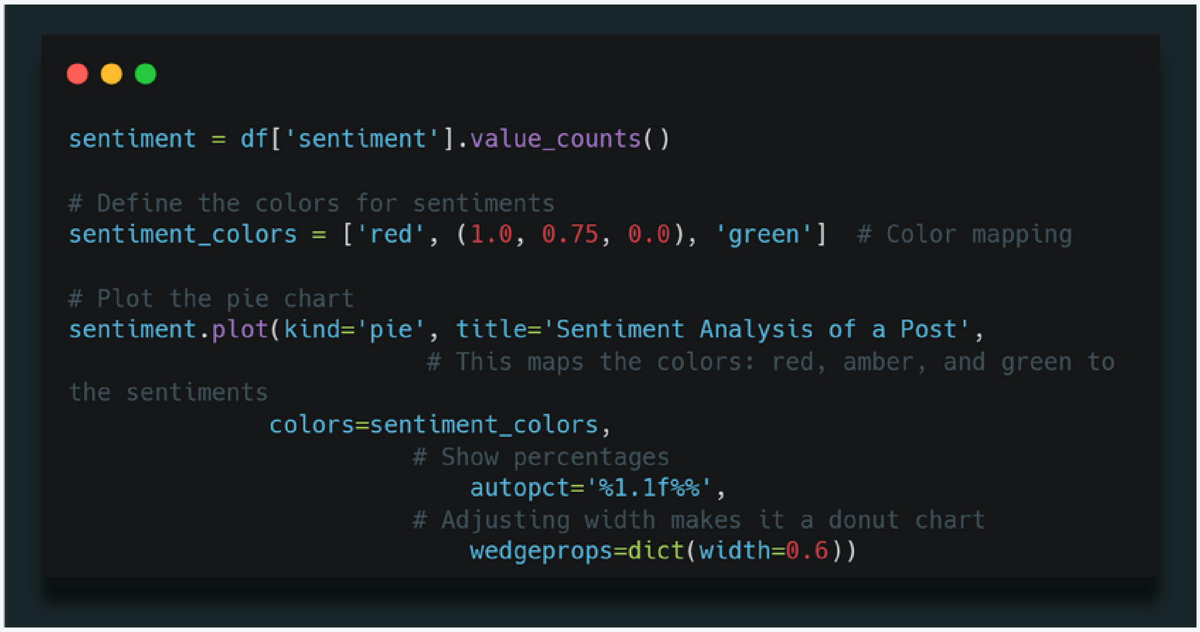
```
## negative comments
df[df['sentiment']== 'negative'].head(10)

#Textblob is good for simple sentiment analysis,
#however, sometime makes mistakes.
```



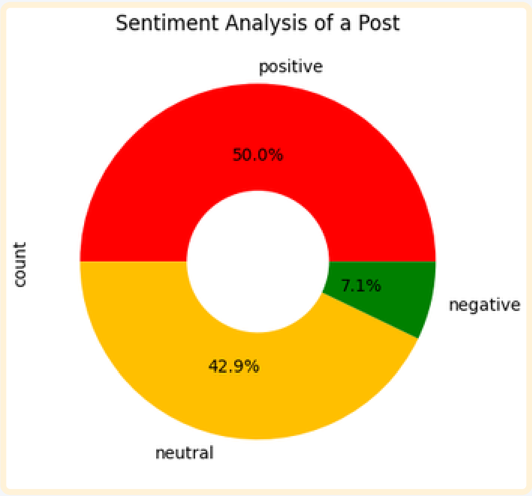
	comment text	polarity	sentiment
7	Gracias por compartir; precisamente estoy estu...	-1.00000	negative
25	Well done for covering a broad base and the re...	-0.46875	negative

SWIPE

Let's show it on bar chart

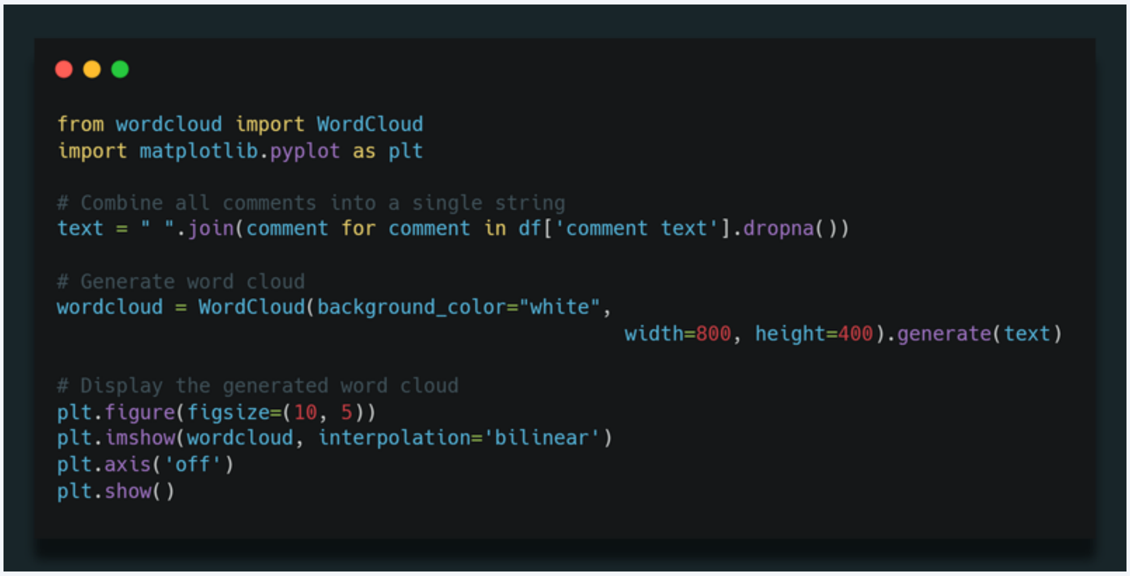


```
sentiment = df['sentiment'].value_counts()

# Define the colors for sentiments
sentiment_colors = ['red', (1.0, 0.75, 0.0), 'green']  # Color mapping

# Plot the pie chart
sentiment.plot(kind='pie', title='Sentiment Analysis of a Post',
                         # This maps the colors: red, amber, and green to
the sentiments
              colors=sentiment_colors,
                        # Show percentages
                            autopct='%1.1f%%',
                        # Adjusting width makes it a donut chart
                            wedgeprops=dict(width=0.6))
```

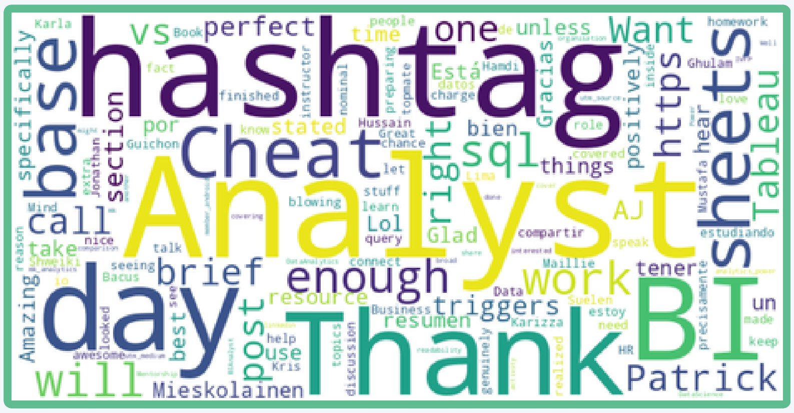


SWIPE

Generate a Word Cloud



```python
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Combine all comments into a single string
text = " ".join(comment for comment in df['comment text'].dropna())

# Generate word cloud
wordcloud = WordCloud(background_color="white",
                                                 width=800, height=400).generate(text)


# Display the generated word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



SWIPE

Conclusion

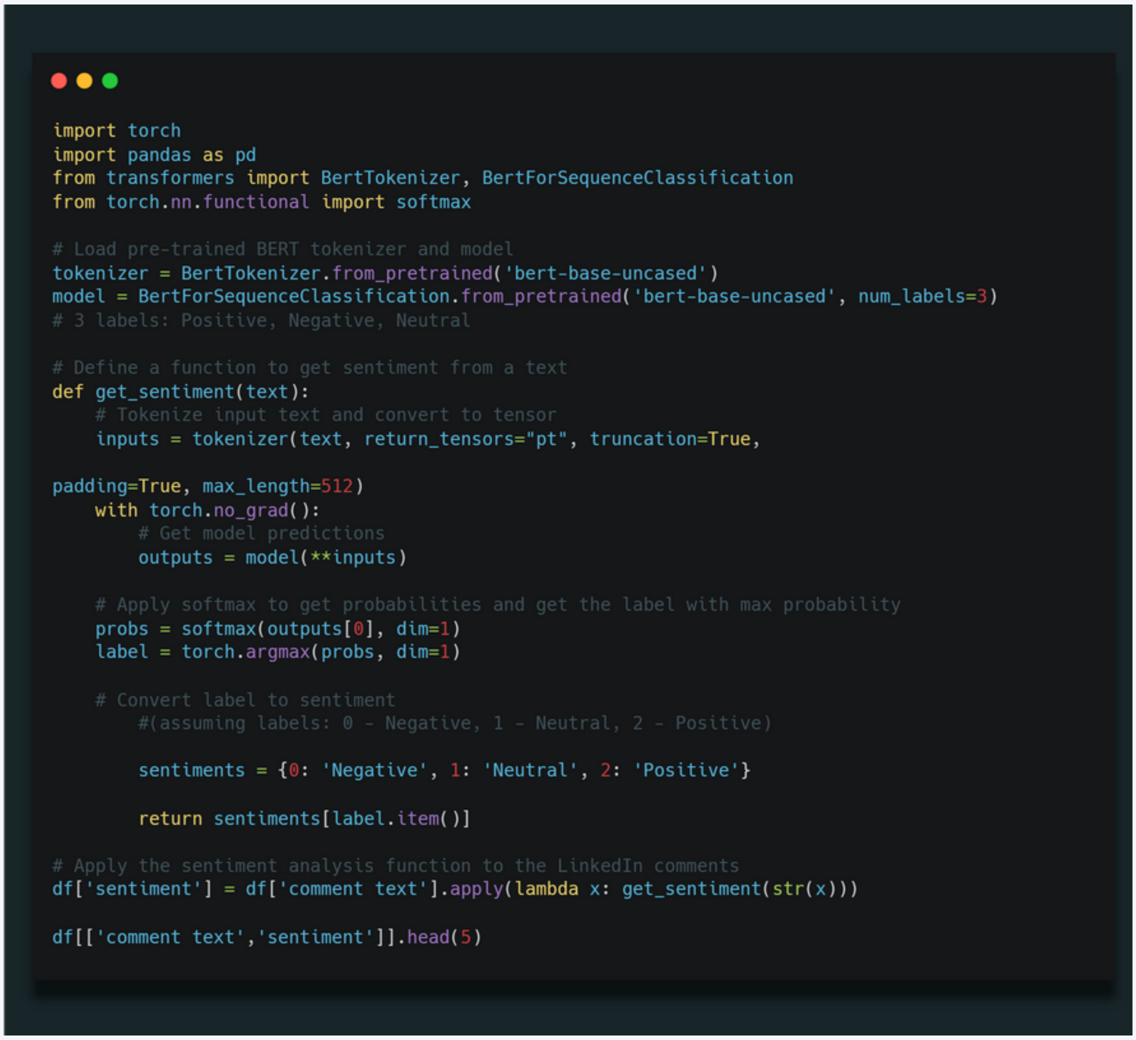
By now, you should have a clear idea about the sentiments of the comments on your LinkedIn post and the most frequently used words in those comments.
This is a basic guide to sentiment analysis and word cloud generation.

For more accurate sentiment analysis, you might consider using more sophisticated models or libraries, training on domain-specific data, or incorporating additional preprocessing steps.

SWIPE

Appendix

Using BERT for better sentiment analysis



```python
import torch
import pandas as pd
from transformers import BertTokenizer, BertForSequenceClassification
from torch.nn.functional import softmax

# Load pre-trained BERT tokenizer and model
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=3)
# 3 labels: Positive, Negative, Neutral

# Define a function to get sentiment from a text
def get_sentiment(text):
    # Tokenize input text and convert to tensor
    inputs = tokenizer(text, return_tensors="pt", truncation=True,

padding=True, max_length=512)
    with torch.no_grad():
        # Get model predictions
        outputs = model(**inputs)

    # Apply softmax to get probabilities and get the label with max probability
    probs = softmax(outputs[0], dim=1)
    label = torch.argmax(probs, dim=1)

    # Convert label to sentiment
        #(assuming labels: 0 - Negative, 1 - Neutral, 2 - Positive)

        sentiments = {0: 'Negative', 1: 'Neutral', 2: 'Positive'}

        return sentiments[label.item()]

# Apply the sentiment analysis function to the LinkedIn comments
df['sentiment'] = df['comment text'].apply(lambda x: get_sentiment(str(x)))

df[['comment text','sentiment']].head(5)
```

SWIPE

Manoj Kumar

Follow to Learn more..

Please like, comment and share with others