Social_media_sentiment_analysis

```
import tweepy
import re
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.sentiment import SentimentIntensityAnalyzer
from textblob import TextBlob
import matplotlib.pyplot as plt


# Set up Twitter API credentials
consumer_key = 'your_consumer_key'
consumer_secret = 'your_consumer_secret'
access_token = 'your_access_token'
access_token_secret = 'your_access_token_secret'


# Set up Tweepy API object
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)


# Collect tweets using a specific hashtag or keyword
tweets = api.search(q='#python', count=100)


# Extract tweet text and store in a list
tweet_texts = [tweet.text for tweet in tweets]


# Remove punctuation and convert to lowercase
tweet_texts = [re.sub(r'[^\w\s]', '', tweet_text.lower()) for tweet_text in tweet_texts]


# Tokenize tweets
tokenized_tweets = [word_tokenize(tweet_text) for tweet_text in tweet_texts]
```

```
# Remove stop words
stop_words = set(stopwords.words('english'))
tokenized_tweets = [[word for word in tweet if word not in stop_words] for tweet in
tokenized_tweets]


# Join tokenized tweets back into strings
tweet_texts = [' '.join(tweet) for tweet in tokenized_tweets]


# Initialize sentiment analyzer
sia = SentimentIntensityAnalyzer()


# Analyze sentiment using NLTK
sentiments = [sia.polarity_scores(tweet_text) for tweet_text in tweet_texts]


# Analyze sentiment using TextBlob
sentiments_tb = [TextBlob(tweet_text).sentiment.polarity for tweet_text in tweet_texts]


# Define a function to classify sentiment
def classify_sentiment(score):
if score > 0.5:
return 'positive'
elif score < -0.5:
return 'negative'
else:
return 'neutral'


# Classify sentiments using NLTK
classified_sentiments = [classify_sentiment(sentiment['compound']) for sentiment in
sentiments]


# Classify sentiments using TextBlob
classified_sentiments_tb = [classify_sentiment(sentiment) for sentiment in sentiments_tb]


# Plot sentiment distribution using NLTK
```

```
plt.hist(classified_sentiments, bins=['positive', 'negative', 'neutral'])
plt.xlabel('Sentiment')
plt.ylabel('Frequency')
plt.title('Sentiment Analysis Results (NLTK)')
plt.show()


# Plot sentiment distribution using TextBlob
plt.hist(classified_sentiments_tb, bins=['positive', 'negative', 'neutral'])
plt.xlabel('Sentiment')
plt.ylabel('Frequency')
plt.title('Sentiment Analysis Results (TextBlob)')
plt.show()
```