Sentiment Analysis Using Python

One of the application of text mining is sentiment analysis. Most of the data is getting generated in textutal format and in the pas few years. Improvemnt is a conntinuous process and many product based companies levrage these text mining techniques to examine the sentiments of the customers to find about what they can improve in the product . This information also helps them to understand the trend and demand of the end user which results in Customer satisfaction .

As text mining is a vast concept , the srticle is divided into two subchapters. The main fpcus of this article will be calculating two scores: sentiment polarity and subjectivity using python. The range of polarity is from -1 to 10 (negative to positive) and will tell us if the text contains positive or negative feedback.Most companies prefer to stop their analysis here but in our second article we will try to extend our analysis by creating some lebels out of these scores.

Import Library

In [2]:
```
import pandas as pd
import re
import string
import numpy as np
import random
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from wordcloud import WordCloud
from textblob import TextBlob
```

Import the Data and Convert the sample data to a csv file

In [8]:
```
# Import the data
df = pd.read_json('E:\Sample Data.txt' , lines = True)

# Convert the sample data to a csv file

df.to_csv('E:\Sample Data.csv',index=None)
```

In [9]:
```
df.head()
```

Out[9]:

	_type	url	date	
0	snscrape.modules.twitter.Tweet	https://twitter.com/theworkingboat/status/1460...	2021-11-16 09:54:20+00:00	
1	snscrape.modules.twitter.Tweet	https://twitter.com/WeHoLove/status/1460523193...	2021-11-16 08:20:39+00:00	#(
2	snscrape.modules.twitter.Tweet	https://twitter.com/bdsrated/status/1460518587...	2021-11-16 08:02:21+00:00	W be\
3	snscrape.modules.twitter.Tweet	https://twitter.com/maggietranquila/status/146...	2021-11-16 07:34:44+00:00	
4	snscrape.modules.twitter.Tweet	https://twitter.com/BadalonaCC/status/14605071...	2021-11-16 07:16:46+00:00	(

5 rows × 28 columns

Data Preprocessing

Now we will perform various pre-processing steps on the dataset that mainly dealt with removing stopwords, removing emojis, The text document is then converted into the lowercase for better generalization.

Subsequently , the punctuations will be cleaned and removed there by reducing the unnecessary noise from the dataset. After that, we will also remove the repeating characters from the words along with removing the URLs as they do not have any significant importance.

At last, we will then perform stemming (reducing the words to their derived stems) and Lemmatizations(reducing the derived words to their root from known as lemma) for a better results.

Data Cleaning

In [10]:
```
df.fillna('', inplace = True)
```

In [11]:
```
df.shape
```

Out[11]:
```
(5642, 28)
```

In [13]:
```
import nltk
from nltk.stem import WordNetLemmatizer
lemma = WordNetLemmatizer()
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\meanu\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Making Statement text in Lower Case

In [14]:
```
df['content'] = df['content'].str.lower()
df['content'].head()
```

Out[14]:
```
0    ⭐ thank you ⭐\n\nour popular #sunday night pub...
1    ahhh big stars in #gayweho #redressparty #mick...
2    without god\nour week would be\nsinday\nmournd...
3    #awesome #attitude #motivation #commitment #su...
4    @badalonacc #sunday training with @omaree02\n\...
Name: content, dtype: object
```

Cleaning and Removing the above stop words list from text

In [15]:
```
STOPWORDS = set(stopwords.words('english'))
def cleaning_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])
df['content'] = df['content'].apply(lambda text: cleaning_stopwords(text))
df['content'].head()
```

Out[15]:
```
0    ⭐ thank ⭐ popular #sunday night pub quizzes hu...
1    ahhh big stars #gayweho #redressparty #mickysw...
2    without god week would sinday mournday tearsda...
3    #awesome #attitude #motivation #commitment #su...
4    @badalonacc #sunday training @omaree02 @cricke...
Name: content, dtype: object
```

Removing Punctuation, Number,and Special Characters

This will replace everything except characters and hastags with spaces."[^a-zA-Z#]" this regular expression means everything except alphabets and hastags.

Cleaning and Removing punctuations

In [19]:
```
import string
english_punctuations = string.punctuation
punctuations_list = english_punctuations
def cleaning_punctuations(text):
    translator = str.maketrans('', '', punctuations_list)
    return text.translate(translator)
df['content']= df['content'].apply(lambda x: cleaning_punctuations(x))
df['content'].head()
```

Out[19]:
```
0    ⭐ thank ⭐ popular sunday night pub quizzes hug...
1    ahhh big stars gayweho redressparty mickysweho...
2    without god week would sinday mournday tearsda...
3    awesome attitude motivation commitment sunday ...
4    badalonacc sunday training omaree02 cricketpun...
Name: content, dtype: object
```

Cleaning and Removing repeating characters

In [22]:
```
def cleaning_repeating_char(text):
    return re.sub(r'(.)1+', r'1',text)
```

```
df['content'] = df['content'].apply(lambda x: cleaning_repeating_char(x))
df['content'].head()
```

Out[22]:
```
0    ⭐ thank ⭐ popular sunday night pub quizzes hug...
1    ahhh big stars gayweho redressparty mickysweho...
2    without god week would sinday mournday tearsda...
3    awesome attitude motivation commitment sunday ...
4    badalonacc sunday training omaree02 cricketpun...
Name: content, dtype: object
```

Cleaning and Removing URLs

In [27]:
```
def cleaning_URLs(data):
    return re.sub('((www.[^s]+)|(https?://[^s]+))',' ',data)
df['content'] = df['content'].apply(lambda x: cleaning_URLs(x))
df['content'].head()
```

Out[27]:
```
0    ⭐ thank ⭐ popular sunday night pub quizzes hug...
1    ahhh big stars gayweho redressparty mickysweho...
2    without god week would sinday mournday tearsda...
3    awesome attitude motivation commitment sunday ...
4    badalonacc sunday training omaree02 cricketpun...
Name: content, dtype: object
```

Cleaning and Removing Numeric numbers

In [29]:
```
def cleaning_numbers(data):
    return re.sub('[0-9]+', ' ', data)
df['content'] = df['content'].apply(lambda x: cleaning_numbers(x))
df['content'].head()
```

Out[29]:
```
0    ⭐ thank ⭐ popular sunday night pub quizzes hug...
1    ahhh big stars gayweho redressparty mickysweho...
2    without god week would sinday mournday tearsda...
3    awesome attitude motivation commitment sunday ...
4    badalonacc sunday training omaree  cricketpunt...
Name: content, dtype: object
```

Remove Short words

We remove those words which are of little or no use. So , we will select the length of words which we want to remove

In [31]:
```
def transform_text(text):
    return ' '.join([word for word in text.split() if len(word) > 2])
df['content'] = df['content'].apply(lambda x: transform_text(x))
df['content'].head()
```

Out[31]:
```
0    thank popular sunday night pub quizzes huge su...
1    ahhh big stars gayweho redressparty mickysweho...
2    without god week would sinday mournday tearsda...
3    awesome attitude motivation commitment sunday ...
4    badalonacc sunday training omaree cricketpuntc...
Name: content, dtype: object
```

Tokenization

Tokenization is a way to split into a list of words.In this example you'll use the Natural Language Toolkit which has built - in functions for tokenization. We can also use regex to tokenize it but it is a bit difficult .Through it gives you more control over our text

Getting tokenization of tweet text

In [33]:
```
# Function which directly tokenize the tweet data :-

from nltk.tokenize import TweetTokenizer

tt= TweetTokenizer()
df['content'] = df['content'].apply(tt.tokenize)
df['content'].head()
```

Out[33]:
```
0    [thank, popular, sunday, night, pub, quizzes, ...
1    [ahhh, big, stars, gayweho, redressparty, mick...
2    [without, god, week, would, sinday, mournday, ...
3    [awesome, attitude, motivation, commitment, su...
4    [badalonacc, sunday, training, omaree, cricket...
Name: content, dtype: object
```

Applying Stemming

In [36]:
```
import nltk
st = nltk.PorterStemmer()
def stemming_on_text(data):
    text = [st.stem(word) for word in data]
    return data
df['content']= df['content'].apply(lambda x: stemming_on_text(x))
df['content'].head()
```

Out[36]:
```
0    [thank, popular, sunday, night, pub, quizzes, ...
1    [ahhh, big, stars, gayweho, redressparty, mick...
2    [without, god, week, would, sinday, mournday, ...
3    [awesome, attitude, motivation, commitment, su...
4    [badalonacc, sunday, training, omaree, cricket...
Name: content, dtype: object
```

Applying Lemmatizer:-

In [37]:
```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\meanu\AppData\Roaming\nltk_data...
```

Out[37]:
```
True
```

In [42]:
```
lm = nltk.WordNetLemmatizer()
def lemmatizer_on_text(data):
    text = [lm.lemmatize(word) for word in data]
    return data
df['content'] = df['content'].apply(lambda x: lemmatizer_on_text(x))
df['content'].head()
```

Out[42]:
```
0    [thank, popular, sunday, night, pub, quizzes, ...
1    [ahhh, big, stars, gayweho, redressparty, mick...
2    [without, god, week, would, sinday, mournday, ...
3    [awesome, attitude, motivation, commitment, su...
4    [badalonacc, sunday, training, omaree, cricket...
Name: content, dtype: object
```

Subjectivity and Polarity

In [45]:
```
# Create a function to get the subjectivity
def getSubjectivity (text):
    # Join the list of words into a single string using a space separator
    text = ' '.join(text)
    return TextBlob(text).sentiment.subjectivity

# Create a function to get the polarity
def getpolarity(text):
    # join the list of words into a single string using a space separator
    text = ' '.join(text)
    return TextBlob(text).sentiment.polarity

# Create two new columns
df['subjectivity'] = df['content'].apply(getSubjectivity)
df['polarity'] = df['content'].apply(getpolarity)

# Show the new dataframe with the new columns
df.head()
```

Out[45]:

	_type	url	date	
0	snscrape.modules.twitter.Tweet	https://twitter.com/theworkingboat/status/1460...	2021-11-16 09:54:20+00:00	r
1	snscrape.modules.twitter.Tweet	https://twitter.com/WeHoLove/status/1460523193...	2021-11-16 08:20:39+00:00	rec
2	snscrape.modules.twitter.Tweet	https://twitter.com/bdsrated/status/1460518587...	2021-11-16 08:02:21+00:00	mc
3	snscrape.modules.twitter.Tweet	https://twitter.com/maggietranquila/status/146...	2021-11-16 07:34:44+00:00	cor
4	snscrape.modules.twitter.Tweet	https://twitter.com/BadalonaCC/status/14605071...	2021-11-16 07:16:46+00:00	[b

5 rows × 30 columns

Compute the negative , neutral and positive analysis

In [49]:
```
#create a function to compute the negative, neutral and positive analysis
def getAnalysis(score):
    if score<0:
        return 'negative'
    elif score==0:
        return 'neutral'
    else:
        return 'positive'

df['analysis']=df['polarity'].apply(getAnalysis)

#show dataFrame
df.head()
```

Out[49]:

	_type	url	date	
0	snscrape.modules.twitter.Tweet	https://twitter.com/theworkingboat/status/1460...	2021-11-16 09:54:20+00:00	
1	snscrape.modules.twitter.Tweet	https://twitter.com/WeHoLove/status/1460523193...	2021-11-16 08:20:39+00:00	rec
2	snscrape.modules.twitter.Tweet	https://twitter.com/bdsrated/status/1460518587...	2021-11-16 08:02:21+00:00	mc
3	snscrape.modules.twitter.Tweet	https://twitter.com/maggietranquila/status/146...	2021-11-16 07:34:44+00:00	cor
4	snscrape.modules.twitter.Tweet	https://twitter.com/BadalonaCC/status/14605071...	2021-11-16 07:16:46+00:00	[b

5 rows × 31 columns

In [50]:
```
# Create two new dataframe all of the positive text
df_positive = df[df['analysis'] == 'positive']

#Create two new dataframe all of the negative text
df_negative = df[df['analysis']== 'nagative']

#create two new dataframe all of the neutral text
df_neutral=df[df['analysis']== 'neutral']
```

Count the number of Positive,Negative,Neutral Reviews

In [51]:
```
tb_counts = df.analysis.value_counts()
tb_counts
```

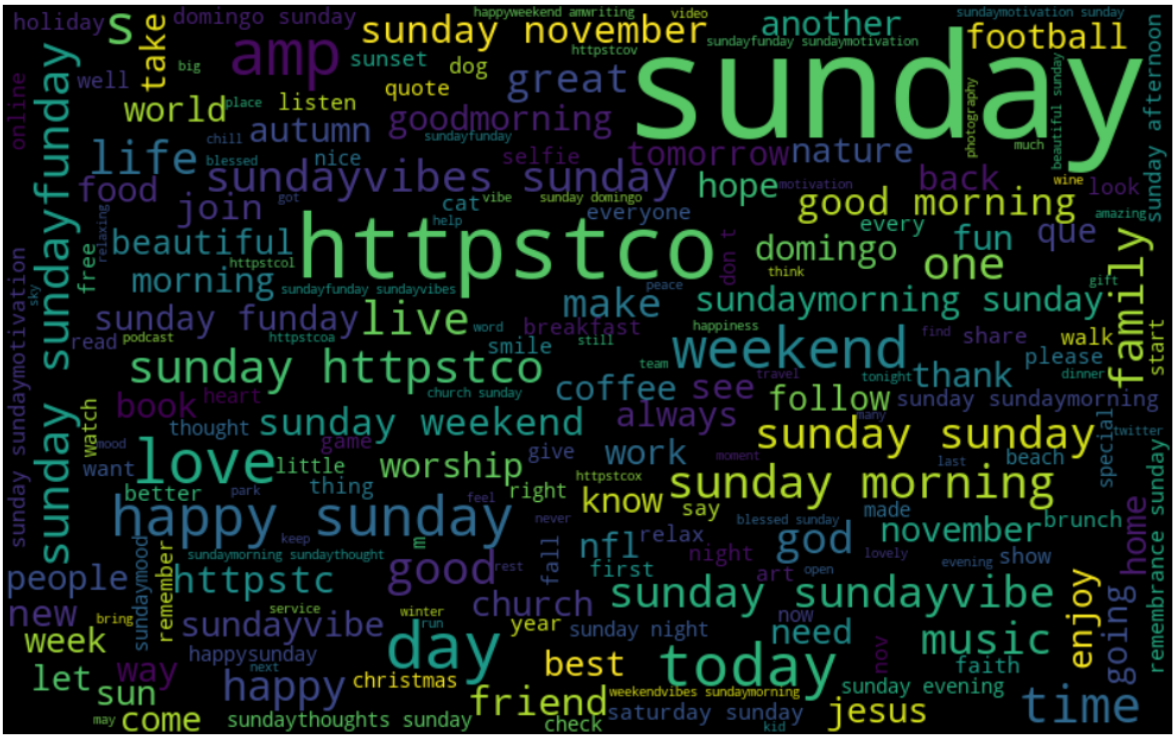
Out[51]:
```
neutral     2695
positive    2524
negative     423
Name: analysis, dtype: int64
```

Data Exploration

Let's form a WordCloud

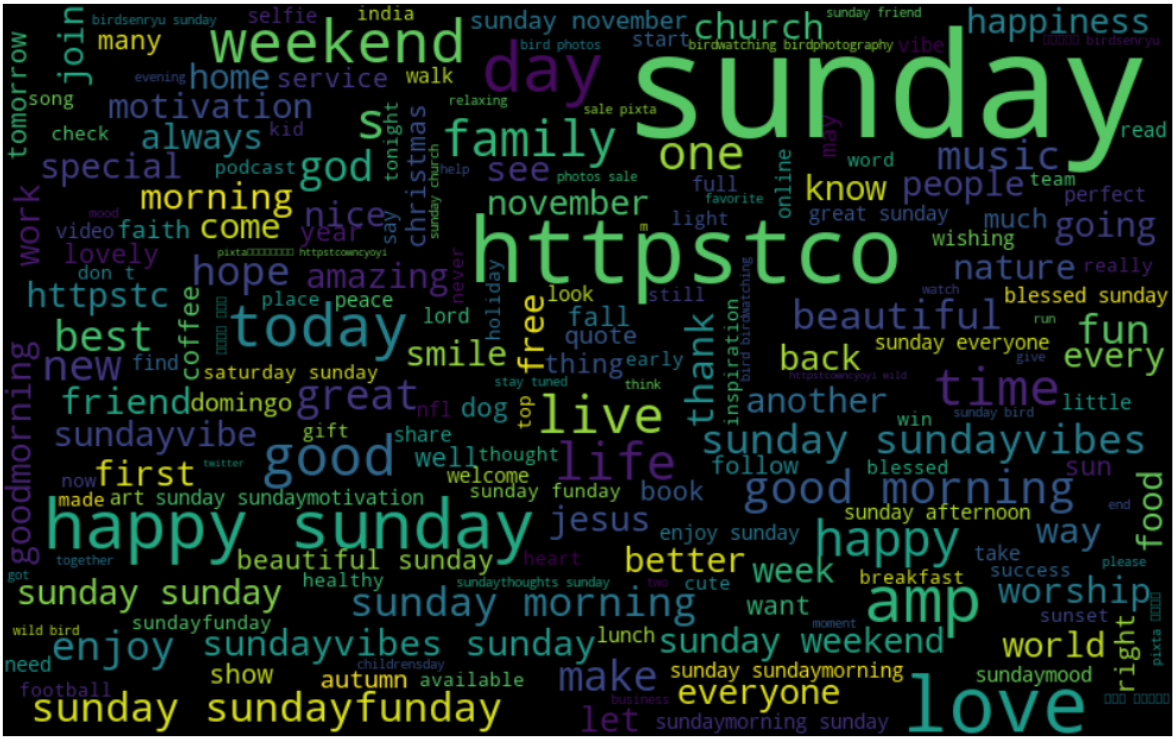
A wordcloud is a visualization wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes

In [53]:
```
# Visualization al tweets

all_words = " ".join(" ".join(sent) for sent in df['content'])

from wordcloud import WordCloud
wordcloud = WordCloud(width=800,height = 500 , random_state = 42, max_font_size =

plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



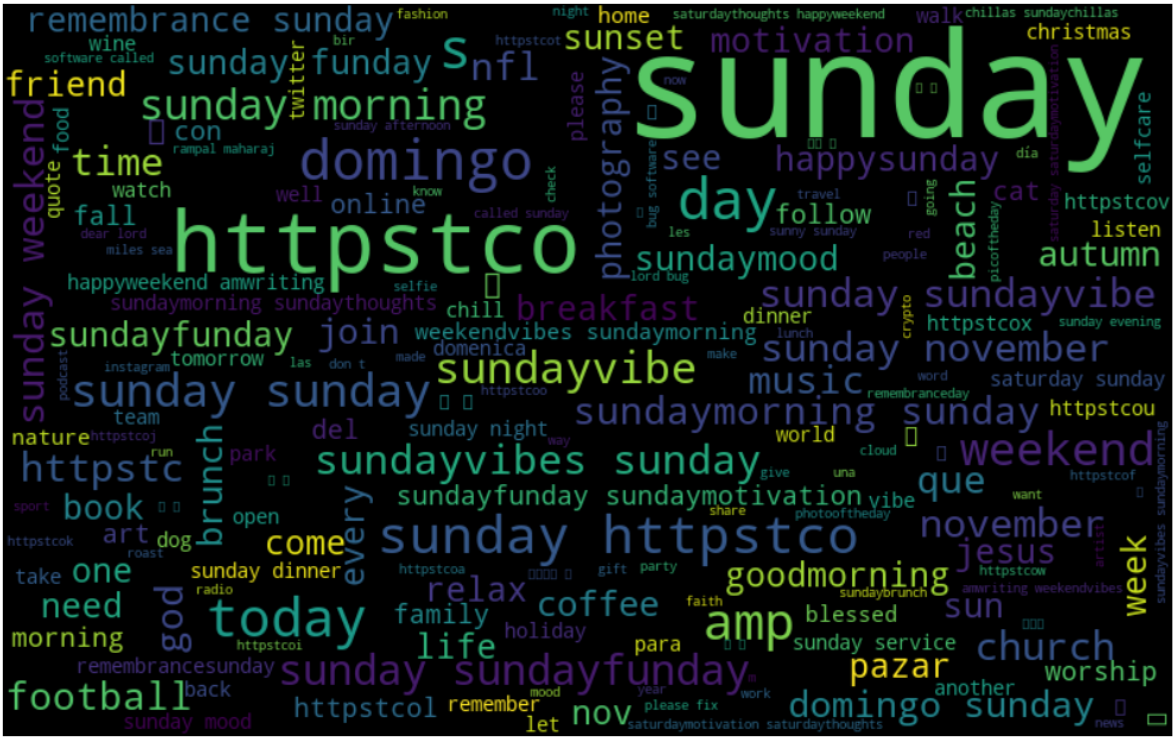
Positive tweets

In [55]:
```
# Visualizing all positive tweets

all_pos_words = " ".join(" ".join(sent) for sent in df_positive['content'])

from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500,random_state=42, max_font_size=100).ge

plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

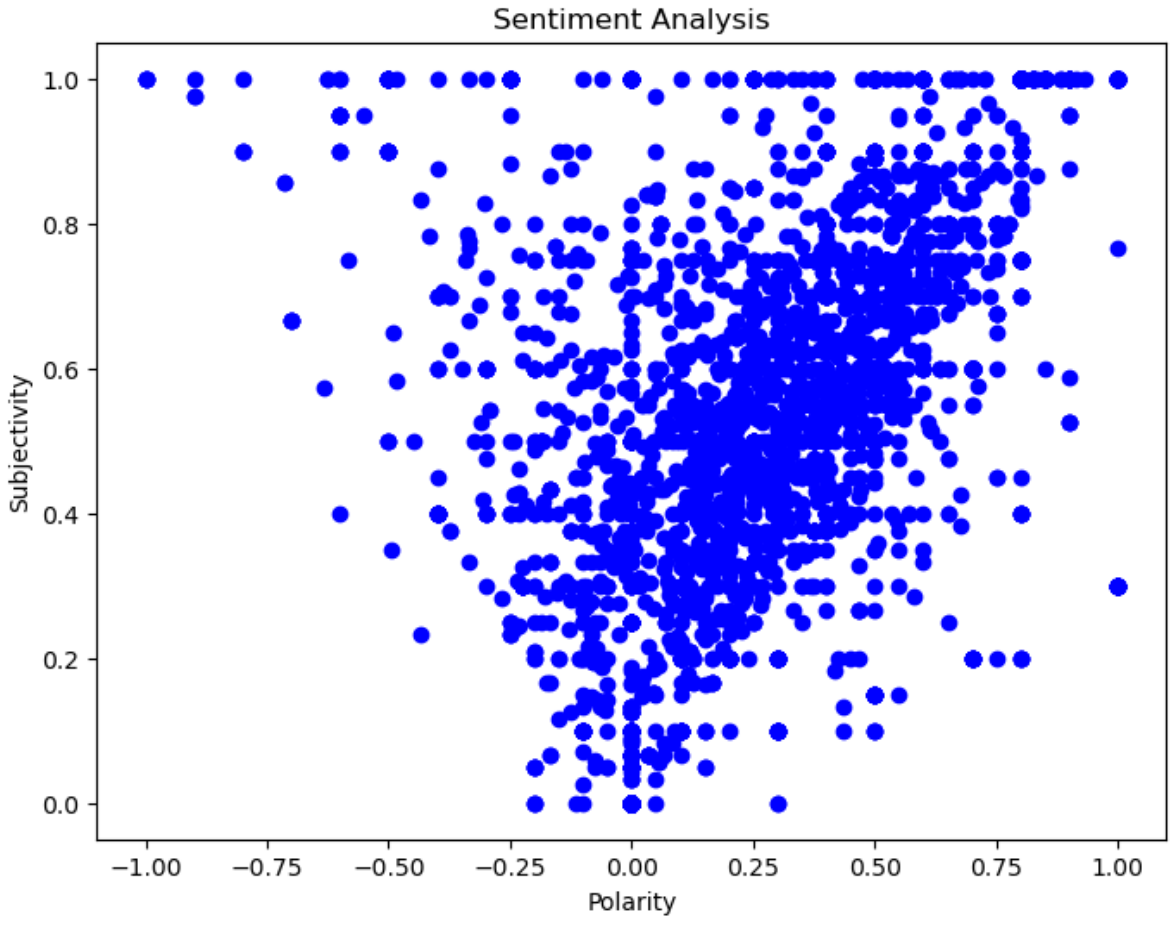


Neutral Tweets

In [66]:
```
# Visualizing all neutral tweets

all_neu_words = " ".join(" ".join(sent) for sent in df_neutral['content'])

from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).g

plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [71]:
```
#plot the polarity and subjectivity
plt.figure(figsize=(8,6))
plt.scatter(df['polarity'],df['subjectivity'],color='blue')
plt.title('Sentiment Analysis')
plt.xlabel('Polarity')
plt.ylabel('Subjectivity')
plt.show()
```

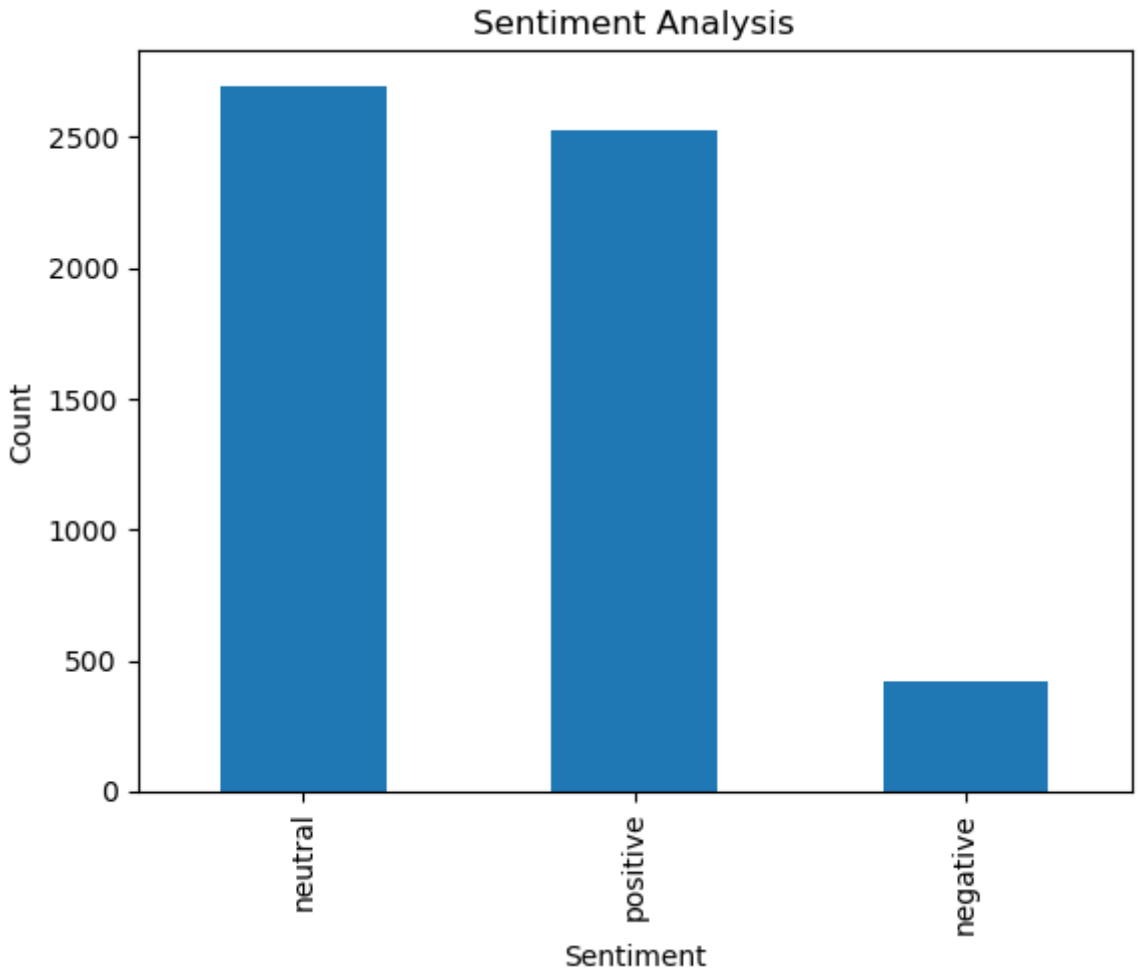


In [72]:
```
# Get the percentage of positive tweets
print("Positive tweets",round((df_positive.shape[0]/df.shape[0])*100,1),"%")
```

```
# Get the percentage of negative of negative tweets
print("Negative tweets",round((df_negative.shape[0]/df.shape[0])*100,1),"%")

#Get the percentage of neutral tweets
print("Neutral tweets",round((df_neutral.shape[0]/df.shape[0])*100,1),"%")
```

```
Positive tweets 44.7 %
Negative tweets 0.0 %
Neutral tweets 47.8 %
```

In [73]:
```
# Show the value counts

df['analysis'].value_counts()

# plot and visualize the counts

plt.title('Sentiment Analysis')
plt.xlabel('Sentiment')
plt.ylabel('Count')
df['analysis'].value_counts().plot(kind='bar')
plt.show()
```



Conclusion

We can see that the maximum percentage of neutral tweets 47.8%, minimum percentage of negative tweets 7.5% and Avg percentage of positive tweets 44.7%.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: