TASK-4

ANALYZE AND VISUALIZE SENTIMENT PATTERNS IN SOCIAL MEDIA DATA TO UNDERSTAND PUBLIC OPINION AND ATTITUDES TOWARDS SPECIFIC TOPICS OR BRANDS.

In [38]:

```
## DATA ##
import numpy as np
import pandas as pd
import re

## NLP ##
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

## Visualization ##
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from plotly.subplots import make_subplots
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff

## ML Modelling ##
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
```

In [39]:

```
col_names = ['ID', 'Entity', 'Sentiment', 'Content']
train_df = pd.read_csv('D:\\mlworld\\twitter_training.csv', names=col_names)
test_df = pd.read_csv('D:\\mlworld\\twitter_validation.csv', names=col_names)
```

In [40]:

```
train_df
```

Out[40]:

	ID	Entity	Sentiment	Content
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
...	...	...	...	...
74677	9200	Nvidia	Positive	Just realized that the Windows partition of my...
74678	9200	Nvidia	Positive	Just realized that my Mac window partition is ...
74679	9200	Nvidia	Positive	Just realized the windows partition of my Mac ...
74680	9200	Nvidia	Positive	Just realized between the windows partition of...
74681	9200	Nvidia	Positive	Just like the windows partition of my Mac is l...

74682 rows × 4 columns

In [41]:

```
test_df
```

Out[41]:

	ID	Entity	Sentiment	Content
0	3364	Facebook	Irrelevant	I mentioned on Facebook that I was struggling ...
1	352	Amazon	Neutral	BBC News - Amazon boss Jeff Bezos rejects clai...
2	8312	Microsoft	Negative	@Microsoft Why do I pay for WORD when it funct...
3	4371	CS-GO	Negative	CSGO matchmaking is so full of closet hacking,...
4	4433	Google	Neutral	Now the President is slapping Americans in the...
...	...	...	...	...
995	4891	GrandTheftAuto(GTA)	Irrelevant	⭐ Toronto is the arts and culture capital of ...
996	4359	CS-GO	Irrelevant	tHIS IS ACTUALLY A GOOD MOVE TOT BRING MORE VI...
997	2652	Borderlands	Positive	Today sucked so it’s time to drink wine n play...
998	8069	Microsoft	Positive	Bought a fraction of Microsoft today. Small wins.
999	6960	johnson&johnson	Neutral	Johnson & Johnson to stop selling talc baby po...

1000 rows × 4 columns

In [42]:

```
train_df.isnull().sum()
```

Out[42]:

```
ID             0
Entity         0
Sentiment      0
Content      686
dtype: int64
```

In [43]:

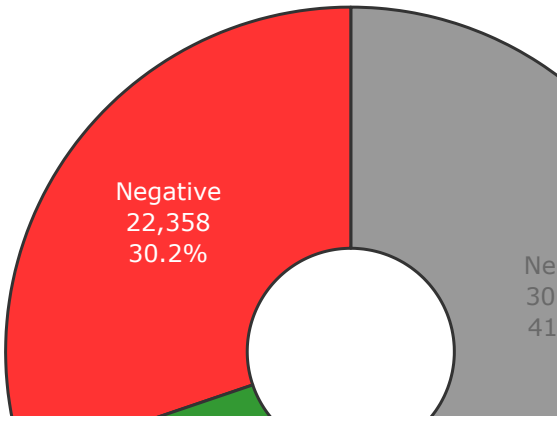
```
#Since there are 686 null values in content (text), I will drop them.

train_df.dropna(subset=['Content'], inplace=True)
train_df['Sentiment'] = train_df['Sentiment'].replace('Irrelevant', 'Neutral')
test_df['Sentiment'] = test_df['Sentiment'].replace('Irrelevant', 'Neutral')
```

In this section, I will first analyze sentiment distribution and sentiment distribution by top 3 entity, and then NLP preprocess texts, and lastly visualize text distribution by each sentiment.

In [44]:

```
sentiment_counts = train_df['Sentiment'].value_counts().sort_index()

sentiment_labels = ['Negative', 'Neutral', 'Positive']
sentiment_colors = ['red', 'grey', 'green']

fig = go.Figure(data=[go.Pie(labels=sentiment_counts.index,
                             values=sentiment_counts.values,
                             textinfo='percent+value+label',
                             marker_colors=sentiment_colors,
                             textposition='auto',
                             hole=.3)])

fig.update_layout(
    title_text='Sentiment Distribution',
    template='plotly_white',
    xaxis=dict(
        title='Sources',
    ),
    yaxis=dict(
        title='Number of Posts in Twitter',
    )
)

fig.update_traces(marker_line_color='black',
                  marker_line_width=1.5,
                  opacity=0.8)

fig.show()
```

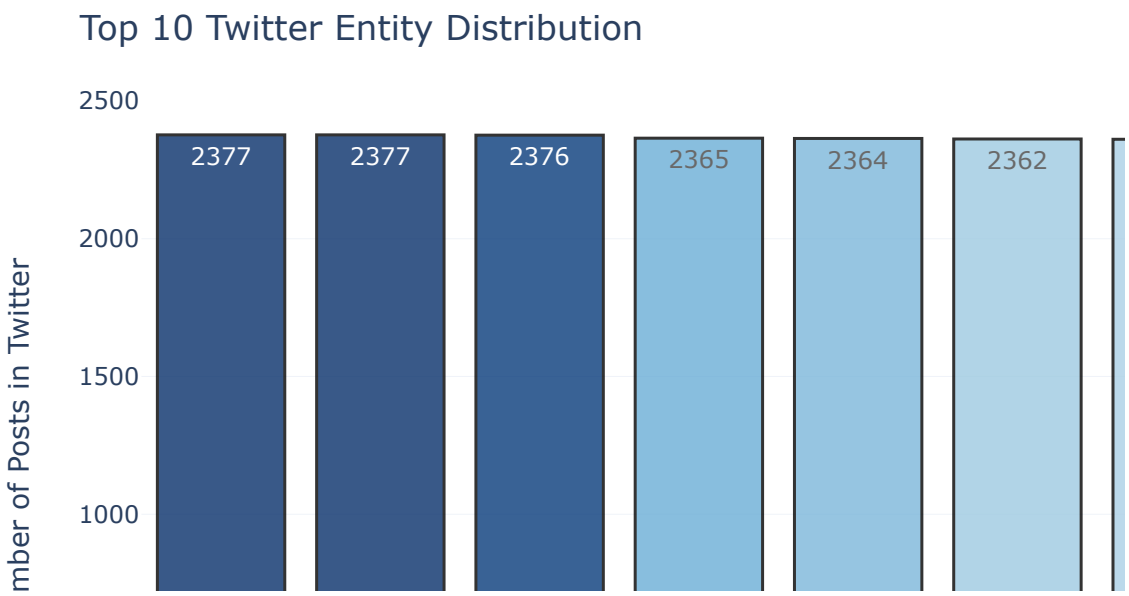
Sentiment Distribution



There are 41.9% of neutral sentiment texts about entity, 30.2% of negative sentiment texts about entity, and 27.9% of positive sentiment texts about entity.

In [45]:

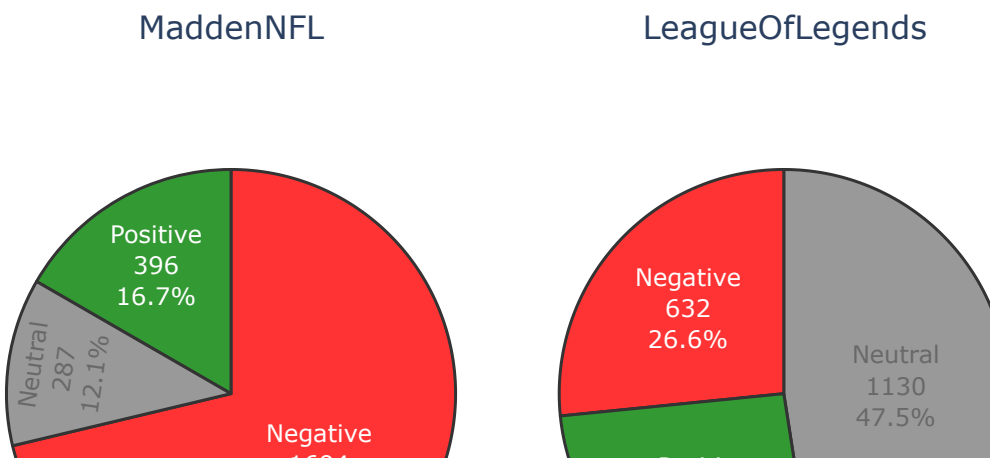
```
top10_entity_counts = train_df['Entity'].value_counts().sort_values(ascending=False)[:10

fig = px.bar(x=top10_entity_counts.index,
             y=top10_entity_counts.values,
             color=top10_entity_counts.values,
             text=top10_entity_counts.values,
             color_continuous_scale='Blues')

fig.update_layout(
    title_text='Top 10 Twitter Entity Distribution',
    template='plotly_white',
    xaxis=dict(
        title='Entity',
    ),
    yaxis=dict(
        title='Number of Posts in Twitter',
    )
)

fig.update_traces(marker_line_color='black',
                  marker_line_width=1.5,
                  opacity=0.8)

fig.show()
```



There are about same amount of data for each entity. MaddenNFL, LeagueOfLegends, CallOfDuty are 3 most distributed entities in the dataset

In [46]:

```
#Sentiment Distribution in Top 3 Entities
top3_entity_df = train_df['Entity'].value_counts().sort_values(ascending=False)[:3]
top3_entity = top3_entity_df.index.tolist()
sentiment_by_entity = train_df.loc[train_df['Entity'].isin(top3_entity)].groupby('Entity

sentiment_labels = ['Negative', 'Neutral', 'Positive']
sentiment_colors = ['red', 'grey', 'green']

row_n = 1
col_n = 3

fig = make_subplots(rows=row_n, cols=col_n,
                    specs=[[{'type':'domain'}, {'type':'domain'}, {'type':'domain'}]],
                    subplot_titles=top3_entity)

for i, col in enumerate(top3_entity):
    fig.add_trace(
        go.Pie(labels=sentiment_labels,
                values=sentiment_by_entity[col].values,
                textinfo='percent+value+label',
                marker_colors=sentiment_colors,
                textposition='auto',
                name=col),
            row=int(i/col_n)+1, col=int(i%col_n)+1)

fig.update_traces(marker_line_color='black',
                  marker_line_width=1.5,
                  opacity=0.8)

fig.show()
```



There are 71.3% negative sentiment tweets about MaddenNFL, 47.5% neutral sentiment tweets about LeagueOfLegends, 44.1% neutral sentiment tweets about CallOfDuty.

Text Analysis with NLP Preprocessing

In this section, I will perform NLP Preprocessing and visualize texts for each sentiment.

Preprocessing Functions Explanations: get_all_string: this function returns all strings in one sentence given a text series get_word: this function returns list of words given a sentence remove_stopword: this function removes stopwords like "the", "is", "and", and etc lemmatize_word: this function lemmatizes the word (i.e. "Caring" --> "Care") create_freq_df: this function returns the frequency dataframe given the list of words

In [47]:

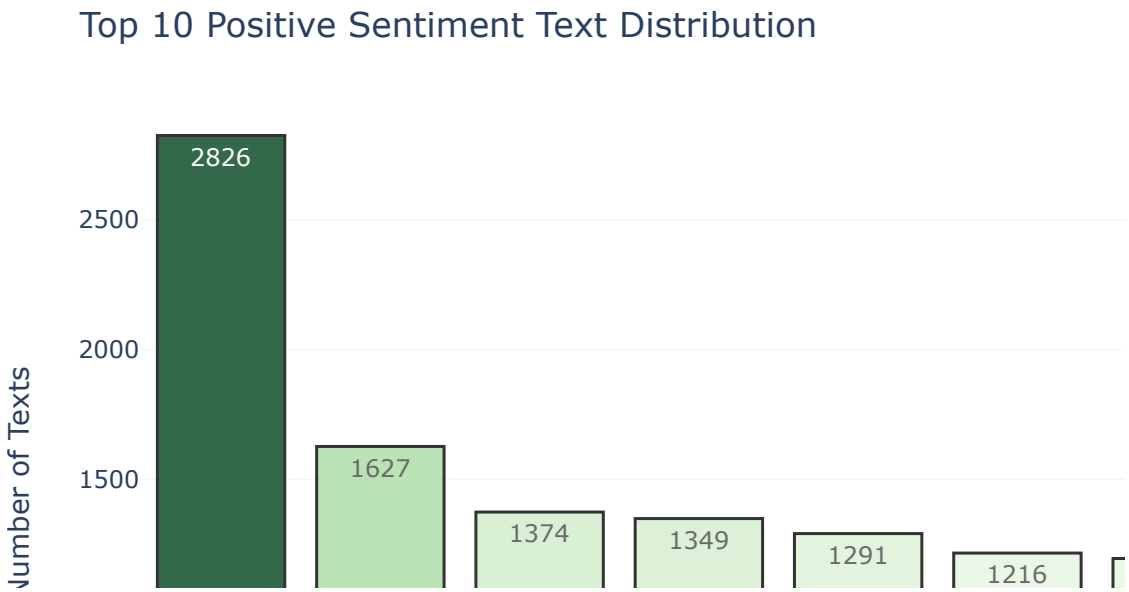
```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
def get_all_string(sentences):
    sentence = ''
    for words in sentences:
        sentence += words
    sentence = re.sub('[^A-Za-z0-9 ]+', '', sentence)
    sentence = re.sub(r'http\S+', '', sentence)
    sentence = sentence.lower()
    return sentence

def get_word(sentence):
    return nltk.RegexpTokenizer(r'\w+').tokenize(sentence)

def remove_stopword(word_tokens):
    stopword_list = stopwords.words('english')
    filtered_tokens = []

    for word in word_tokens:
        if word not in stopword_list:
            filtered_tokens.append(word)
    return filtered_tokens

def lemmatize_words(filtered_tokens):
    lemm = WordNetLemmatizer()
    cleaned_tokens = [lemm.lemmatize(word) for word in filtered_tokens]
    return cleaned_tokens
def create_freq_df(cleaned_tokens):
    fdist = nltk.FreqDist(cleaned_tokens)
    freq_df = pd.DataFrame.from_dict(fdist, orient='index')
    freq_df.columns = ['Frequency']
    freq_df.index.name = 'Term'
    freq_df = freq_df.sort_values(by=['Frequency'], ascending=False)
    freq_df = freq_df.reset_index()
    return freq_df
def preprocess(series):
    all_string = get_all_string(series)
    words = get_word(all_string)
    filtered_tokens = remove_stopword(words)
    cleaned_tokens = lemmatize_words(filtered_tokens)
    return cleaned_tokens
def plot_text_distribution(x_df, y_df, color, title, xaxis_text, yaxis_text):

    fig = px.bar(x=x_df,
                 y=y_df,
                 color=y_df,
                 text=y_df,
                 color_continuous_scale=color)

    fig.update_layout(
        title_text=title,
        template='plotly_white',
        xaxis=dict(
            title=xaxis_text,
        ),
        yaxis=dict(
            title=yaxis_text,
        )
```

```
    )

    fig.update_traces(marker_line_color='black',
                      marker_line_width=1.5,
                      opacity=0.8)

    fig.show()
def create_wordcloud(freq_df, title, color):

    data = freq_df.set_index('Term').to_dict()['Frequency']

    plt.figure(figsize = (20,15))
    wc = WordCloud(width=800,
                height=400,
                max_words=100,
                colormap= color,
                max_font_size=200,
                min_font_size = 1 ,
                random_state=8888,
                background_color='white').generate_from_frequencies(data)

    plt.imshow(wc, interpolation='bilinear')
    plt.title(title, fontsize=20)
    plt.axis('off')
    plt.show()
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\91939\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\91939\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

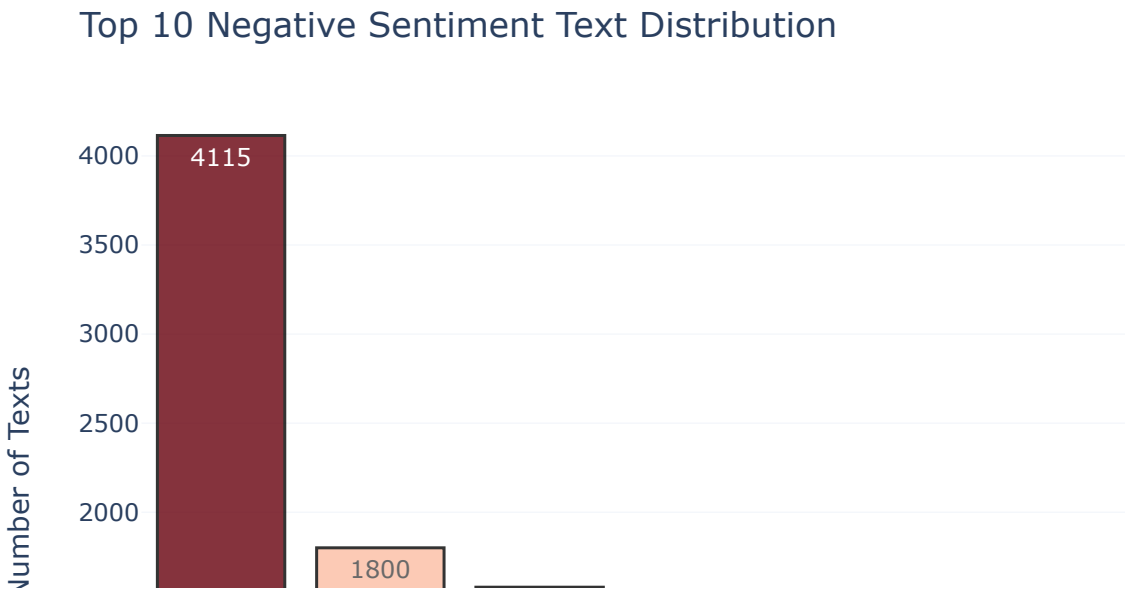
In [48]:

```
positive_words = preprocess(train_df.loc[train_df['Sentiment'] == 'Positive']['Content']
positive_words_df = create_freq_df(positive_words)
top10_positive_words = positive_words_df[:10]

plot_text_distribution(top10_positive_words['Term'], top10_positive_words['Frequency'],
                  'Greens', 'Top 10 Positive Sentiment Text Distribution', 'Text', 'Numb
#create_wordcloud(positive_words_df, 'Positive Sentiment Text Distribution', 'BuGn')
```



In [49]:

```
negative_words = preprocess(train_df.loc[train_df['Sentiment'] == 'Negative']['Content']
negative_words_df = create_freq_df(negative_words)
top10_negative_words = negative_words_df[:10]

plot_text_distribution(top10_negative_words['Term'], top10_negative_words['Frequency'],
                  'Reds', 'Top 10 Negative Sentiment Text Distribution', 'Text', 'Number
#create_wordcloud(negative_words_df, 'Negative Sentiment Text Distribution', 'OrRd')
```



In [ ]:

In [ ]:

In [ ]: