Dutch_Social_Media

December 26, 2022

[1]:
```
import pandas as pd
import json
import re
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from wordcloud import WordCloud
import scipy
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score
import nltk
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
```

[2]:
```
# Read the JSON file into a Pandas dataframe
with open("C:/Users/visha/Downloads/Compressed/dutch_tweets_chunk0.json/
 ↪dutch_tweets_chunk0.json") as input:
    data=json.load(input)
df=pd.DataFrame(data)
```

[3]:
```
# Print the first few rows of the dataset
print(df.head())

# Check the shape of the dataset
print(df.shape)

# Check the data types of the columns
print(df.dtypes)

# Check for missing values
print(df.isnull().sum())
```

```python
# Get some summary statistics for the numerical columns
print(df.describe())

# Check the value counts of the categorical columns
cat=[]
for col in df.select_dtypes(include=["object"]).columns:
    cat.append(df[col].value_counts())
```

```
                                           full_text  \
0  @pflegearzt @Friedelkorn @LAguja44 Pardon, wol…
1  RT @grantshapps: Aviation demand is reduced du…
2  RT @DDStandaard: De droom van D66 wordt werkel…
3  RT @DDStandaard: De droom van D66 wordt werkel…
4  De droom van D66 wordt werkelijkheid: COVID-19…

                                    text_translation     created_at  \
0  @pflegearzt @Friedelkorn @ LAguja44 Pardon wol…  1583756789000
1  RT @grantshapps: Aviation demand is reduced du…  1583756794000
2  RT @DDStandaard: The D66 dream come true: COVI…  1583756797000
3  RT @DDStandaard: The D66 dream come true: COVI…  1583756797000
4  The D66 dream becomes reality: COVID-19 super …  1583756807000

   screen_name                                       description  \
0  TheoRettich  I  science, therefore a Commie.    FALGSC: P…
1  davidiwanow  I tweet a lot but love to engage & converse. P…
2      EricL65                                              None
3      EricL65                                              None
4     EhrErwin  Budget-Life Coach. Time management Coaching. b…

                                     desc_translation  weekofyear  weekday  \
0  I  science, Therefore a Commie.   FALGSC: Par…          11        0
1  I tweet a lot but love to engage and converse…          11        0
2                                                None          11        0
3                                                None          11        0
4  Budget-Life Coach. Time management coaching. h…          11        0

   day  month  …                          point  latitude longitude altitude  \
0    9      3  …  (52.5001698, 5.7480821, 0.0)  52.50017  5.748082      0.0
1    9      3  …  (52.3727598, 4.8936041, 0.0)  52.37276  4.893604      0.0
2    9      3  …                          None       NaN       NaN      0.0
3    9      3  …                          None       NaN       NaN      0.0
4    9      3  …  (52.3727598, 4.8936041, 0.0)  52.37276  4.893604      0.0

        province  hisco_standard  hisco_code industry sentiment_pattern  \
0      Flevoland            None        None    False               0.0
1  Noord-Holland            None        None    False               0.0
2          False            None        None    False               0.0
3          False            None        None    False               0.0
```

```
4  Noord-Holland            None        None    False                   0.0

  subjective_pattern
0                0.0
1                0.0
2                0.0
3                0.0
4                0.0

[5 rows x 23 columns]
(27019, 23)
full_text              object
text_translation       object
created_at              int64
screen_name            object
description            object
desc_translation       object
weekofyear              int64
weekday                 int64
day                     int64
month                   int64
year                    int64
location               object
point_info             object
point                  object
latitude              float64
longitude             float64
altitude              float64
province               object
hisco_standard         object
hisco_code             object
industry                 bool
sentiment_pattern     float64
subjective_pattern    float64
dtype: object
full_text                 0
text_translation          0
created_at                0
screen_name               0
description            4737
desc_translation       4738
weekofyear                0
weekday                   0
day                       0
month                     0
year                      0
location              11746
point_info            13521
```

```
point                  13521
latitude               13521
longitude              13521
altitude                1775
province                 304
hisco_standard         20144
hisco_code             20144
industry                   0
sentiment_pattern          0
subjective_pattern         0
dtype: int64
         created_at    weekofyear       weekday           day         month  \
count  2.701900e+04  27019.000000  27019.000000  27019.000000  27019.000000
mean   1.592197e+12     24.583700      2.555572     14.089937      6.017913
std    5.882117e+09      9.732801      2.063301      9.383028      2.255116
min    1.580012e+12      4.000000      0.000000      1.000000      1.000000
25%    1.587297e+12     16.000000      1.000000      5.000000      4.000000
50%    1.592898e+12     26.000000      2.000000     16.000000      6.000000
75%    1.597772e+12     34.000000      5.000000     23.000000      8.000000
max    1.600207e+12     38.000000      6.000000     31.000000      9.000000

          year      latitude     longitude  altitude  sentiment_pattern  \
count  27019.0  13498.000000  13498.000000   25244.0       27019.000000
mean    2020.0     49.625468      4.544151       0.0           0.037602
std        0.0     11.984083     21.674134       0.0           0.276415
min     2020.0    -79.406307   -157.795990       0.0          -1.000000
25%     2020.0     51.842652      4.686789       0.0           0.000000
50%     2020.0     52.372760      4.898047       0.0           0.000000
75%     2020.0     52.500170      5.748082       0.0           0.125000
max     2020.0     64.145981    176.897763       0.0           1.000000

       subjective_pattern
count        27019.000000
mean             0.376768
std              0.350845
min             -0.300000
25%              0.000000
50%              0.400000
75%              0.675000
max              1.000000
```

[4]:
```
df.columns
```

[4]:
```
Index(['full_text', 'text_translation', 'created_at', 'screen_name',
       'description', 'desc_translation', 'weekofyear', 'weekday', 'day',
       'month', 'year', 'location', 'point_info', 'point', 'latitude',
       'longitude', 'altitude', 'province', 'hisco_standard', 'hisco_code',
```

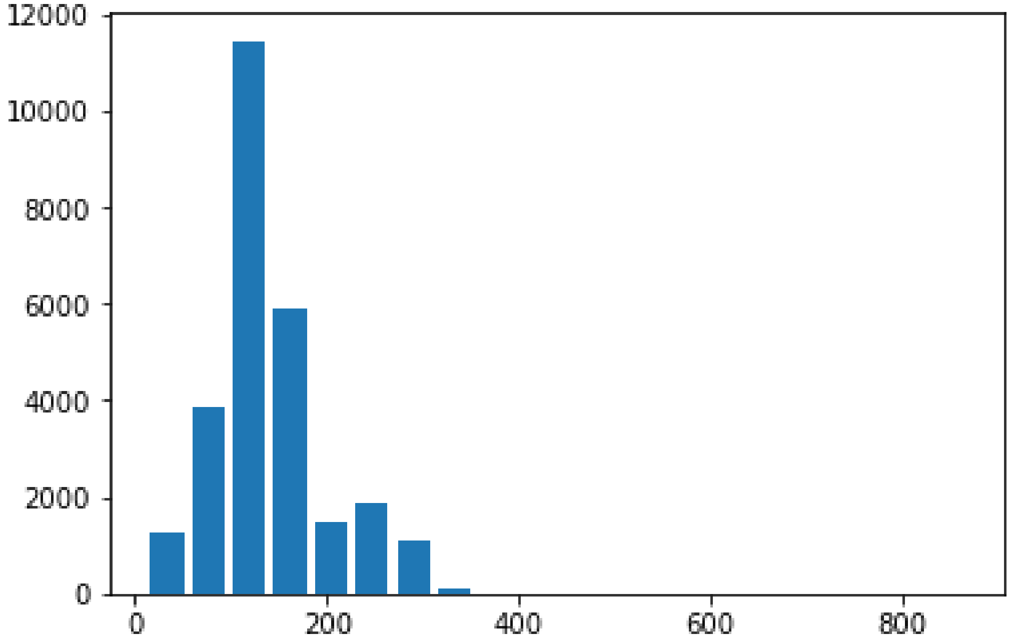
```
       'industry', 'sentiment_pattern', 'subjective_pattern'],
      dtype='object')
```

[5]:
```
#binning continous values to -1,0 & 1
threshold=0
df['sentiment_pattern'] = np.where(df['sentiment_pattern'] < threshold, -1, np.
 ↪where(df['sentiment_pattern'] > threshold, 1, 0))
df['sentiment_pattern'].value_counts()
```

[5]:
```
 0    11743
 1     9230
-1     6046
Name: sentiment_pattern, dtype: int64
```

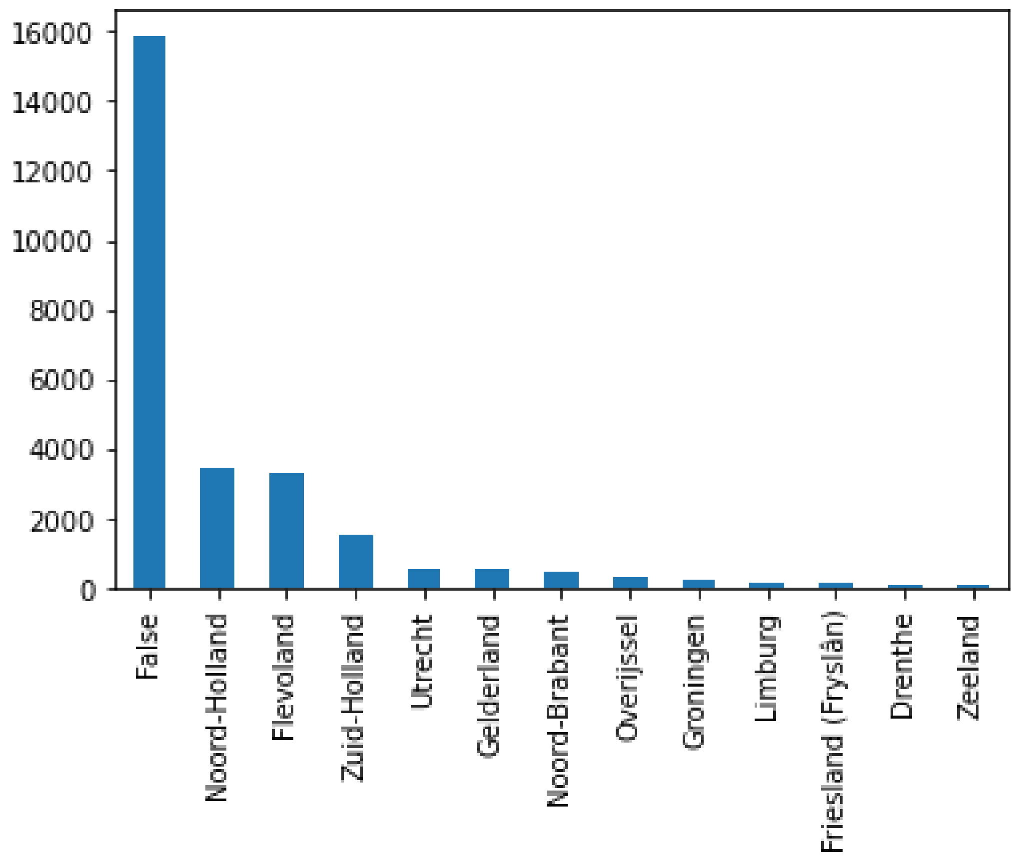
[6]:
```
# Create a Scattergeo map
data = [go.Scattergeo(
    lon = df['longitude'], # Column containing the longitude values
    lat = df['latitude'], # Column containing the latitude values
    text = df['text_translation'], # Column containing the tweet text
    mode = 'markers',
    marker = dict(
        size = 4,
        color = 'red',
        line_color = 'black',
        line_width = 1
    )
)]

# Plot the map
fig = go.Figure(data=data)
fig.show()
```

[7]:
```
# Plot a histogram of the tweet lengths
df['tweet_length'] = df['text_translation'].apply(len)
plt.hist(df['tweet_length'], bins=20,rwidth=0.8)
plt.show()
```



[8]:
```
#Tweet count based on province
df['province'].value_counts().plot.bar()
```

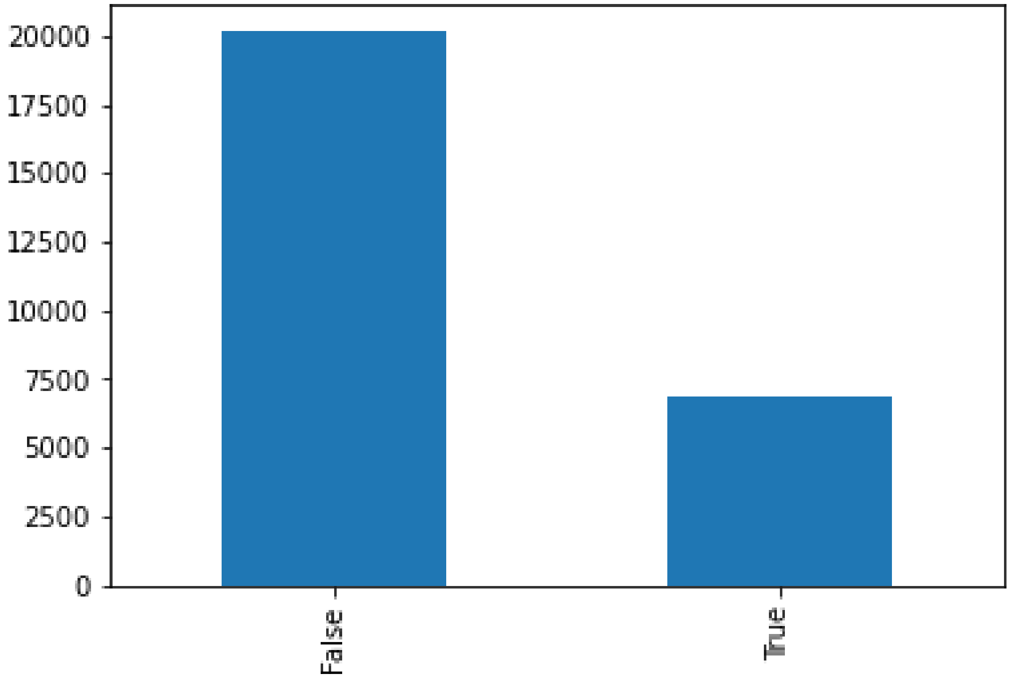
[8]: <AxesSubplot:>



- Most users dont have their location enabled.

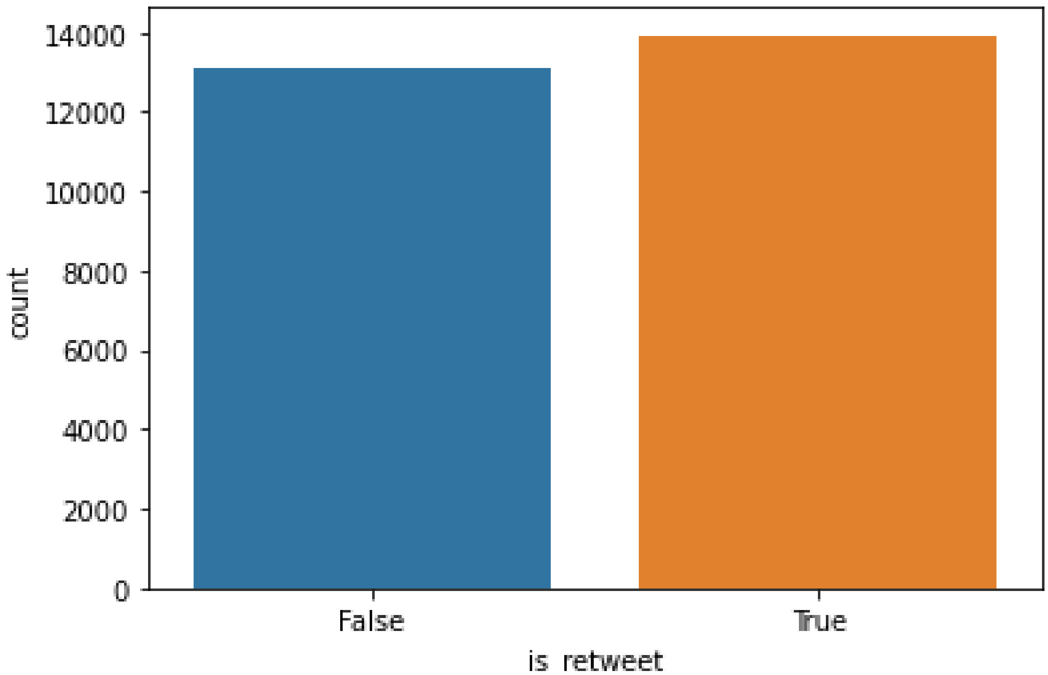
[9]:
```
df['industry'].value_counts().plot.bar()
```

[9]: <AxesSubplot:>



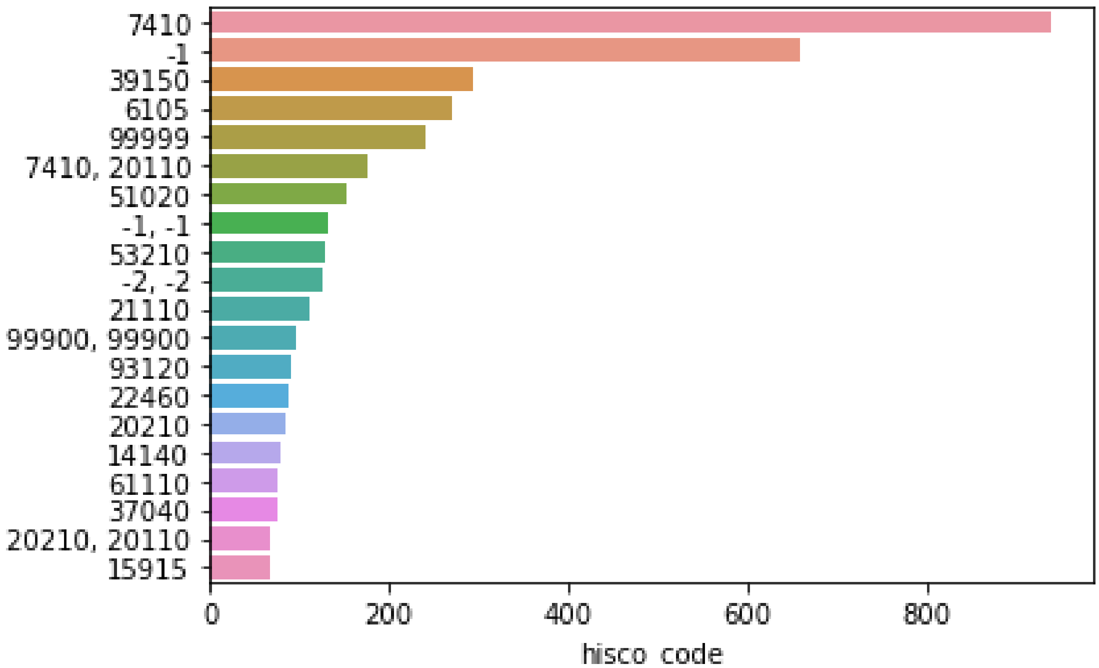
- Industry classification is possible for 30% of the tweets

[10]:
```
# Create a new column indicating whether the tweet is a retweet
df['is_retweet'] = df['text_translation'].str.startswith('RT @')
# Plot a bar plot of the number of tweets that are retweets
sns.countplot(x='is_retweet', data=df)
plt.show()
```



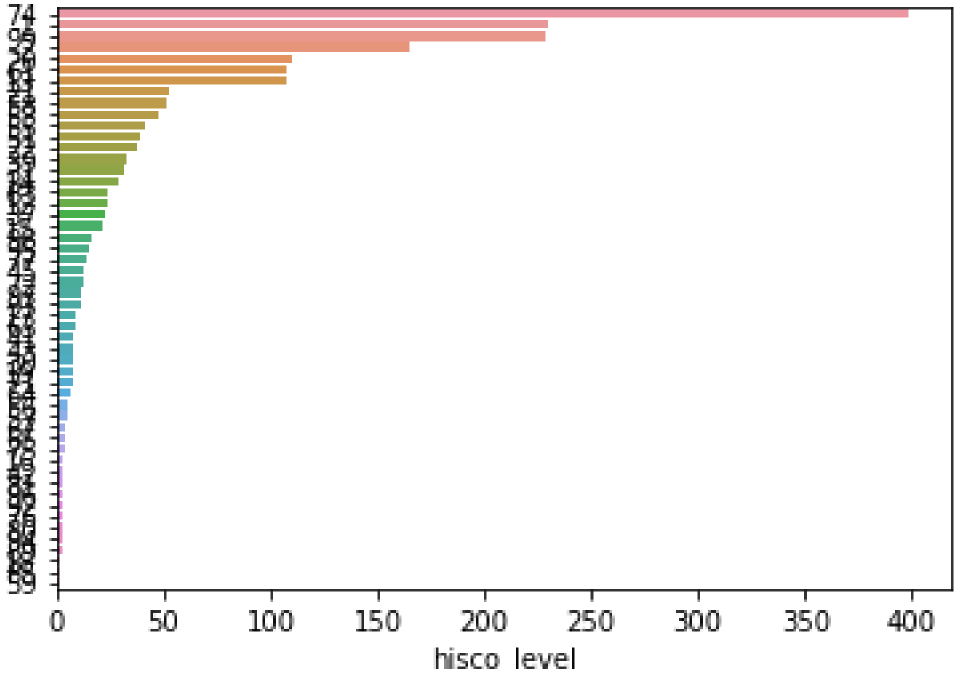
[11]:

```
# Count the number of tweets with a HISCO code
df['hisco_code'].count()

# Print the unique values in the HISCO column
df['hisco_code'].unique()


# Plot a barplot of the most common HISCO codes
top_hisco = df['hisco_code'].value_counts()[:20]
sns.barplot(x=top_hisco, y=top_hisco.index)
plt.show()
```



[ ]:

[12]:
```
# Create a new column with the HISCO code level
df['hisco_level'] = df['hisco_code'].str[:2]

# Plot a barplot of the number of tweets by HISCO level
hisco_levels = df['hisco_level'].value_counts()
sns.barplot(x=hisco_levels, y=hisco_levels.index)
plt.show()
```



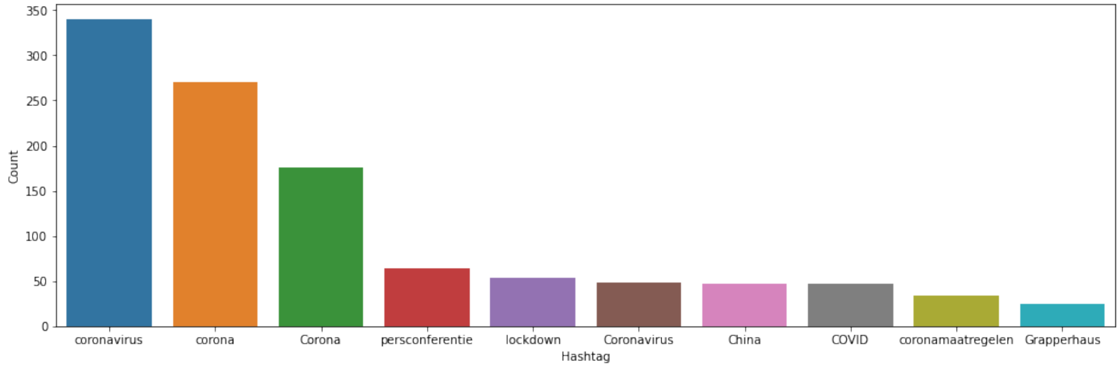
[23]:

```
# Cleaning tweets
def clean_tweet(text):
    Text = ' '
    wordLemm = WordNetLemmatizer()
    temp = text.lower()
    temp = re.sub("rt @[A-Za-z0-9_]+","", temp)
    temp = re.sub("@[A-Za-z0-9_]+","", temp)
    temp = re.sub("#[A-Za-z0-9_]+","", temp)
    temp = re.sub(r'http\S+', '', temp)
    temp = re.sub('[:()!?]', ' ', temp)
    temp = re.sub('\[.*?\]',' ', temp)
    tweetwords=''
    for word in temp.split():
        if len(word)>1:
                    # Lemmatizing the word.
                    word = wordLemm.lemmatize(word)
                    tweetwords+= (word+' ')

    Text+=tweetwords
    return Text
# Extract hashtags
def extract_hashtags(text):
    hashtag_pattern = re.compile(r"#\S+")
    return hashtag_pattern.findall(text)
```

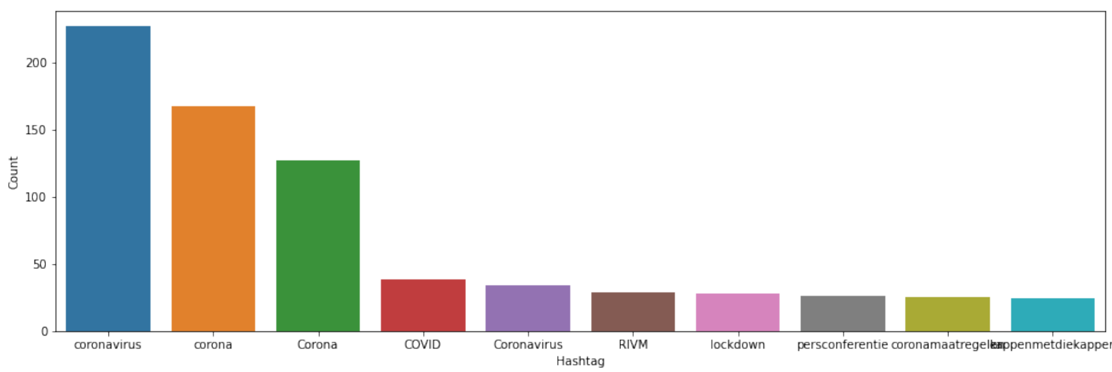
```python
# Extract emojis
def extract_emojis(text):
    emoji_pattern = re.compile("["
        u"\U0001f600-\U0001f64f"  # emoticons
        u"\U0001f300-\U0001f5ff"  # symbols & pictographs
        u"\U0001f680-\U0001f6ff"  # transport & map symbols
        u"\U0001f1e0-\U0001f1ff"  # flags (iOS)
                           "]+", flags=re.UNICODE)
    return emoji_pattern.findall(text)
```

[18]:
```python
# extracting hashtags from non racist/sexist tweets
# collecting the hashtags

def hashtag_extract(x):
    hashtags = []

    for i in x:
        ht = re.findall(r"#(\w+)", i)
        hashtags.append(ht)

    return hashtags

HT_positive =␣
 ↪hashtag_extract(df['text_translation'][df['sentiment_pattern']==1])

# extracting hashtags from racist/sexist tweets
HT_negative =␣
 ↪hashtag_extract(df['text_translation'][df['sentiment_pattern']==-1])

# unnesting list
HT_positive = sum(HT_positive,[])
HT_negative = sum(HT_negative,[])
```

[19]:
```python
#Positive Hashtags
a = nltk.FreqDist(HT_positive)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})
# selecting top 10 most frequent hashtags
d = d.nlargest(columns="Count", n = 10)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```

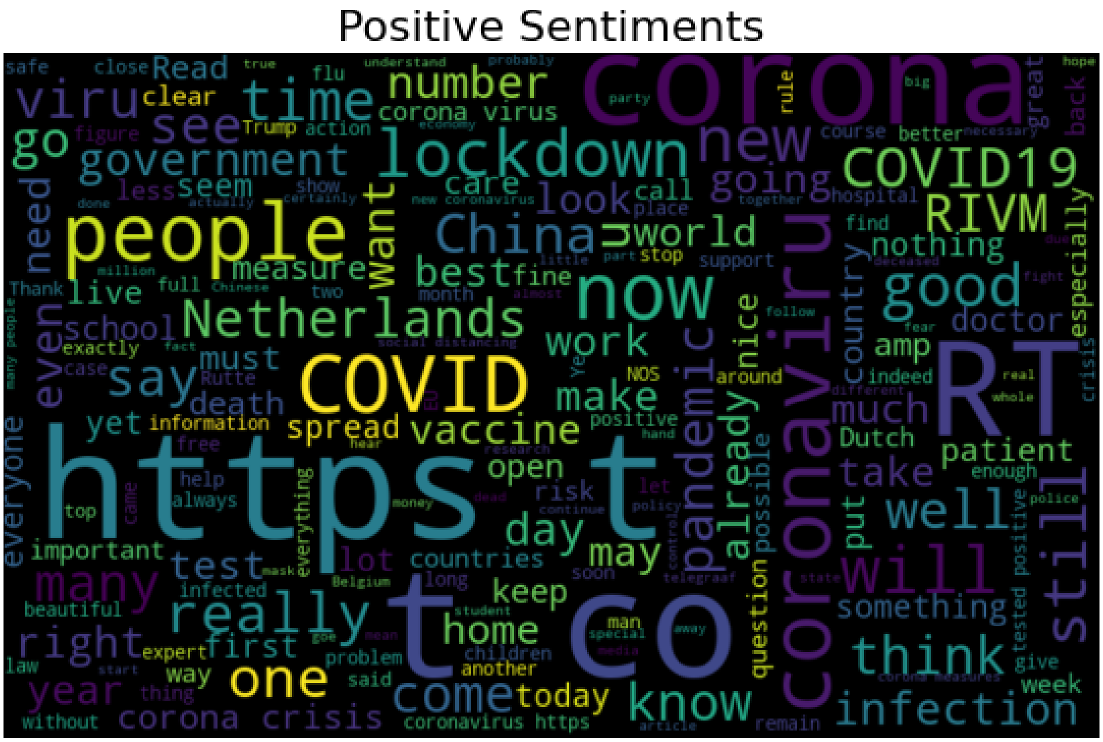


[20]:
```
#Negative Hashtags
b = nltk.FreqDist(HT_negative)
e = pd.DataFrame({'Hashtag': list(b.keys()), 'Count': list(b.values())})
# selecting top 10 most frequent hashtags
e = e.nlargest(columns="Count", n = 10)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=e, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```



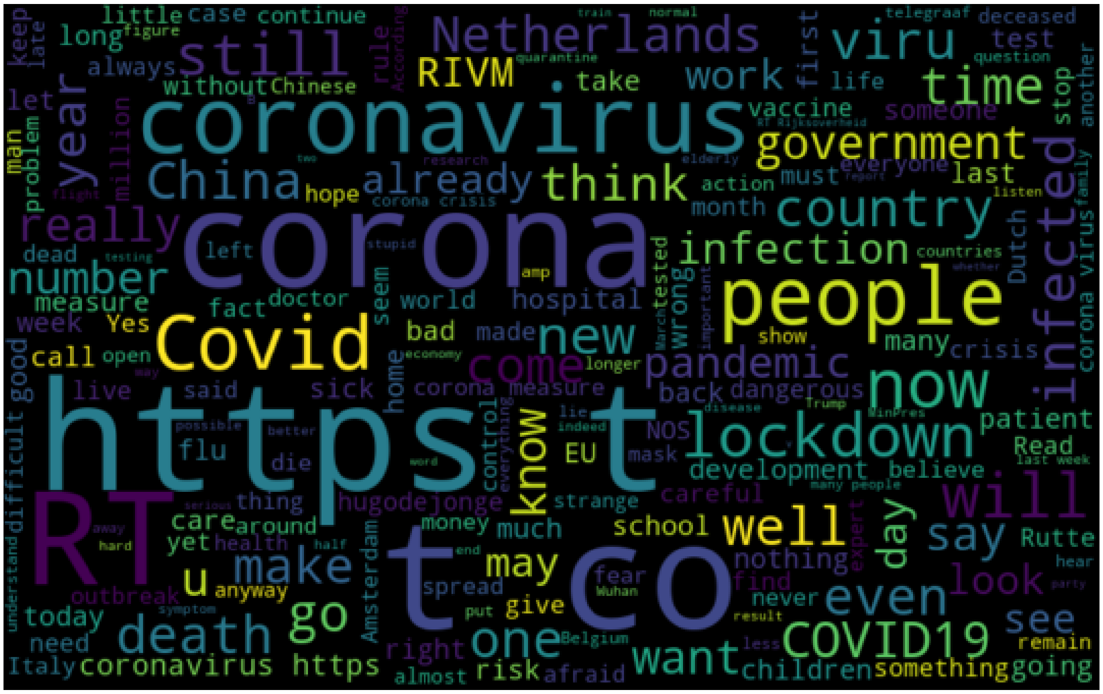
[21]:
```
#Getting all positive sentiments
all_words = ' '.join([text for text in␣
↪df['text_translation'][df['sentiment_pattern']==1]])
wordcloud = WordCloud(width=800, height=500, random_state=21,␣
↪max_font_size=110).generate(all_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("Positive Sentiments", fontsize = 22)
```

```
plt.show()
```



[16]:
```
all_words = ' '.join([text for text in df['text_translation'][df['sentiment_pattern']==-1]])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(all_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("Negative Sentiments", fontsize = 22)
plt.show()
```

Negative Sentiments




```
[24]: # Extract features from the tweet text
vectorizer = TfidfVectorizer()
X_tweet = vectorizer.fit_transform(df["text_translation"].apply(clean_tweet))
```

```
[25]: # Extract features from the hashtags
vectorizer = TfidfVectorizer()
X_hashtags = vectorizer.fit_transform(df["text_translation"].
  ↪apply(extract_hashtags).apply(" ".join))
```

```
[ ]: # Extract features from the emojis
vectorizer = CountVectorizer()
X_emojis = vectorizer.fit_transform(df["text_translation"].
  ↪apply(extract_emojis).apply(" ".join))
```

```
[27]: # Combine the extracted features
X = scipy.sparse.hstack([X_tweet, X_hashtags],)

# Assign the emoji score as the target variable
y = df["sentiment_pattern"]

# Split the data into a training set and a test set
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Scale the data

scaler = StandardScaler(with_mean=False)
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[28]:
```
# Build the model
model = LogisticRegression(max_iter=1000)
model.fit(X_train_scaled, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred,average='macro')
recall = recall_score(y_test, y_pred,average='macro')

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

```
Accuracy: 0.77
Precision: 0.76
Recall: 0.76
```

[ ]:
```

```