Dataset Information

The objective of this task is to detect hate speech in tweets. For the sake of simplicity, we say a tweet contains hate speech if it has a racist or sexist sentiment associated with it. So, the task is to classify racist or sexist tweets from other tweets.

Formally, given a training sample of tweets and labels, where label '1' denotes the tweet is racist/sexist and label '0' denotes the tweet is not racist/sexist, your objective is to predict the labels on the test dataset.

For training the models, we provide a labelled dataset of 31,962 tweets. The dataset is provided in the form of a csv file with each line storing a tweet id, its label and the tweet.

Import modules

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
import nltk
import warnings
%matplotlib inline

warnings.filterwarnings('ignore')
```

Loading the dataset

In [2]:

```
df = pd.read_csv('Twitter Sentiments.csv')
df.head()
```

Out[2]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

In [3]:
```
# datatype info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31962 entries, 0 to 31961
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      31962 non-null  int64 
 1   label   31962 non-null  int64 
 2   tweet   31962 non-null  object
dtypes: int64(2), object(1)
memory usage: 749.2+ KB
```

Preprocessing the dataset

In [4]:
```
# removes pattern in the input text
def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for word in r:
        input_txt = re.sub(word, "", input_txt)
    return input_txt
```

In [5]:
```
df.head()
```

Out[5]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

In [6]:
```
# remove twitter handles (@user)
df['clean_tweet'] = np.vectorize(remove_pattern)(df['tweet'], "@[\w]*")
```

In [7]:
```
df.head()
```

Out[7]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can't use cause th...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation	factsguide: society now #motivation

Loading [MathJax]/extensions/Safe.js

In [8]:
```
# remove special characters, numbers and punctuations
df['clean_tweet'] = df['clean_tweet'].str.replace("[^a-zA-Z#]", " ")
df.head()
```

Out[8]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can t use cause th...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation	factsguide society now #motivation

In [9]:
```
# remove short words
df['clean_tweet'] = df['clean_tweet'].apply(lambda x: " ".join([w for w in x.split() if
df.head()
```

Out[9]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when father dysfunctional selfish drags kids i...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks #lyft credit cause they offer wheelchai...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model love take with time
4	5	0	factsguide: society now #motivation	factsguide society #motivation

In [10]:
```
# individual words considered as tokens
tokenized_tweet = df['clean_tweet'].apply(lambda x: x.split())
tokenized_tweet.head()
```

Out[10]:
```
0    [when, father, dysfunctional, selfish, drags, ...
1    [thanks, #lyft, credit, cause, they, offer, wh...
2                                [bihday, your, majesty]
3                          [#model, love, take, with, time]
4                        [factsguide, society, #motivation]
Name: clean_tweet, dtype: object
```

In [11]:
```
# stem the words
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()

tokenized_tweet = tokenized_tweet.apply(lambda sentence: [stemmer.stem(word) for word in
tokenized_tweet.head()
```

Out[11]:
```
0    [when, father, dysfunct, selfish, drag, kid, i...
1    [thank, #lyft, credit, caus, they, offer, whee...
2                                [bihday, your, majesti]
3                          [#model, love, take, with, time]
4                              [factsguid, societi, #motiv]
Name: clean_tweet, dtype: object
```

In [12]:
```
# combine words into single sentence
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = " ".join(tokenized_tweet[i])

df['clean_tweet'] = tokenized_tweet
df.head()
```

Loading [MathJax]/extensions/Safe.js

Out[12]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when father dysfunct selfish drag kid into dys...
1	2	0	@user @user thanks for #lyft credit i can't us...	thank #lyft credit caus they offer wheelchair ...
2	3	0	bihday your majesty	bihday your majesti
3	4	0	#model i love u take with u all the time in ...	#model love take with time
4	5	0	factsguide: society now #motivation	factsguid societi #motiv

Exploratory Data Analysis

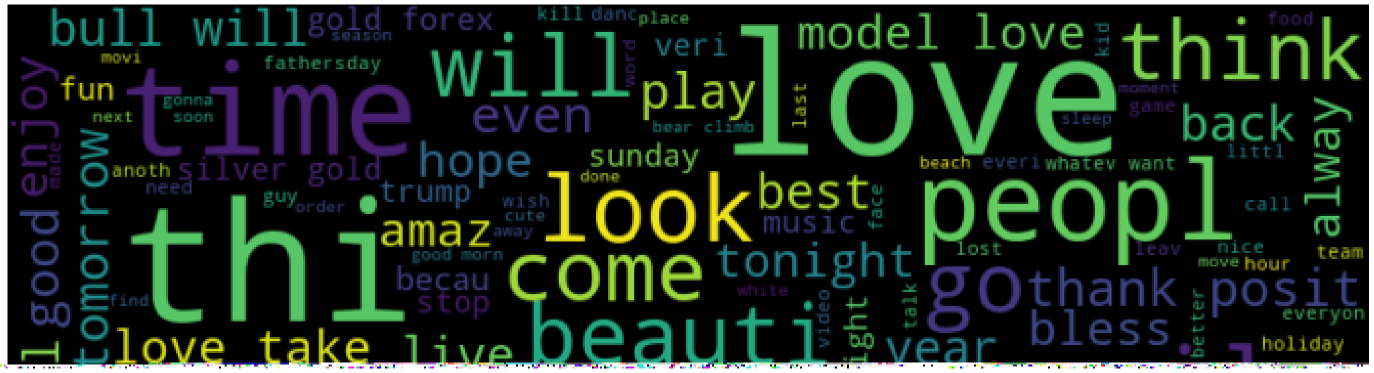
In [14]:
```
# !pip install wordcloud
```

```
Collecting wordcloud
  Downloading wordcloud-1.8.1-cp38-cp38-win_amd64.whl (155 kB)
Requirement already satisfied: pillow in c:\programdata\anaconda3\lib\site-packages (fro
m wordcloud) (7.2.0)
Requirement already satisfied: numpy>=1.6.1 in c:\programdata\anaconda3\lib\site-package
s (from wordcloud) (1.18.5)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages
(from wordcloud) (3.2.2)
Requirement already satisfied: python-dateutil>=2.1 in c:\programdata\anaconda3\lib\site
-packages (from matplotlib->wordcloud) (2.8.1)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-package
s (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\programdat
a\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-pa
ckages (from matplotlib->wordcloud) (1.2.0)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (f
rom python-dateutil>=2.1->matplotlib->wordcloud) (1.15.0)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.8.1
```

In [15]:
```
# visualize the frequent words
all_words = " ".join([sentence for sentence in df['clean_tweet']])

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

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



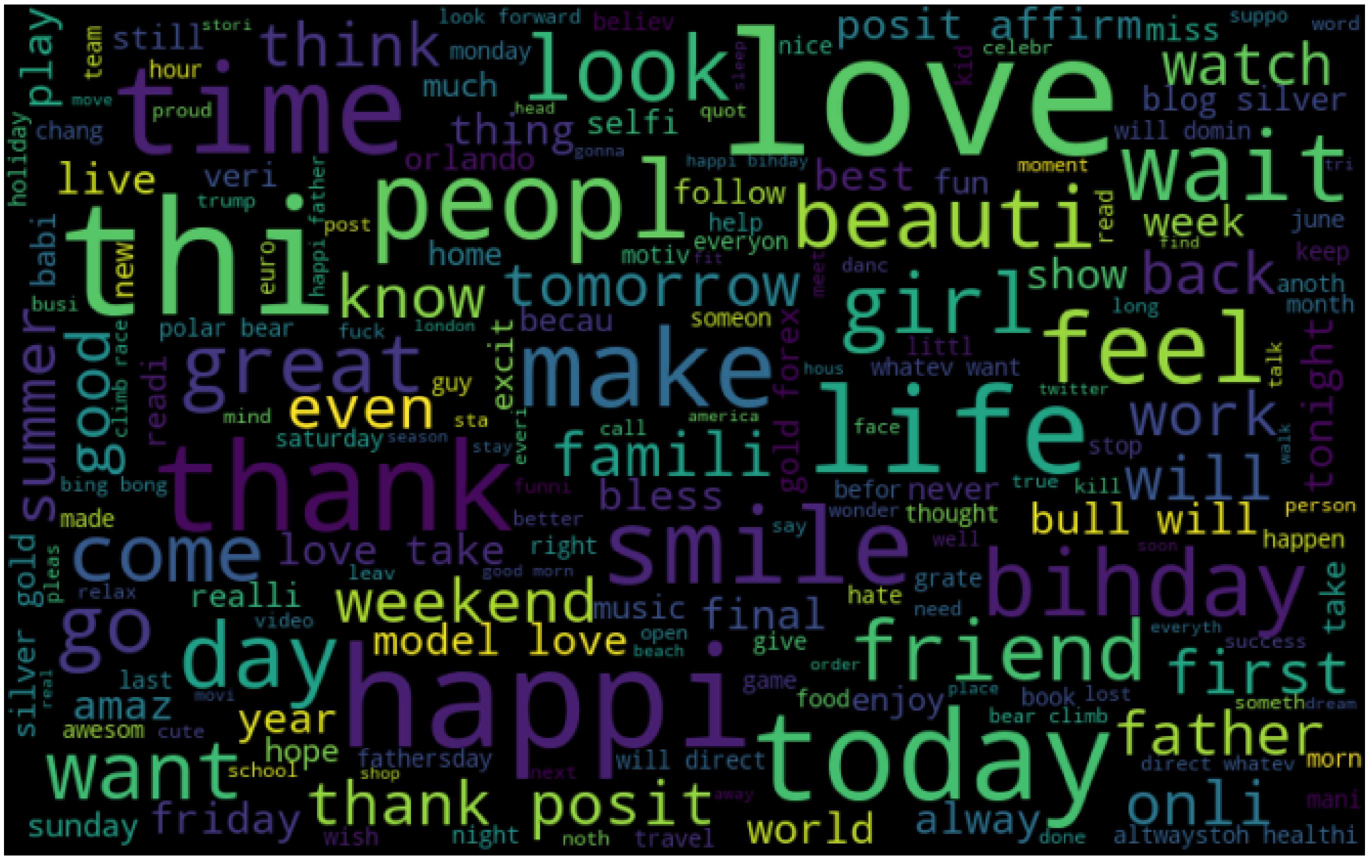
In [16]:

```
# frequent words visualization for +ve
all_words = " ".join([sentence for sentence in df['clean_tweet'][df['label']==0]])

wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).generat

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

Loading [MathJax]/extensions/Safe.js

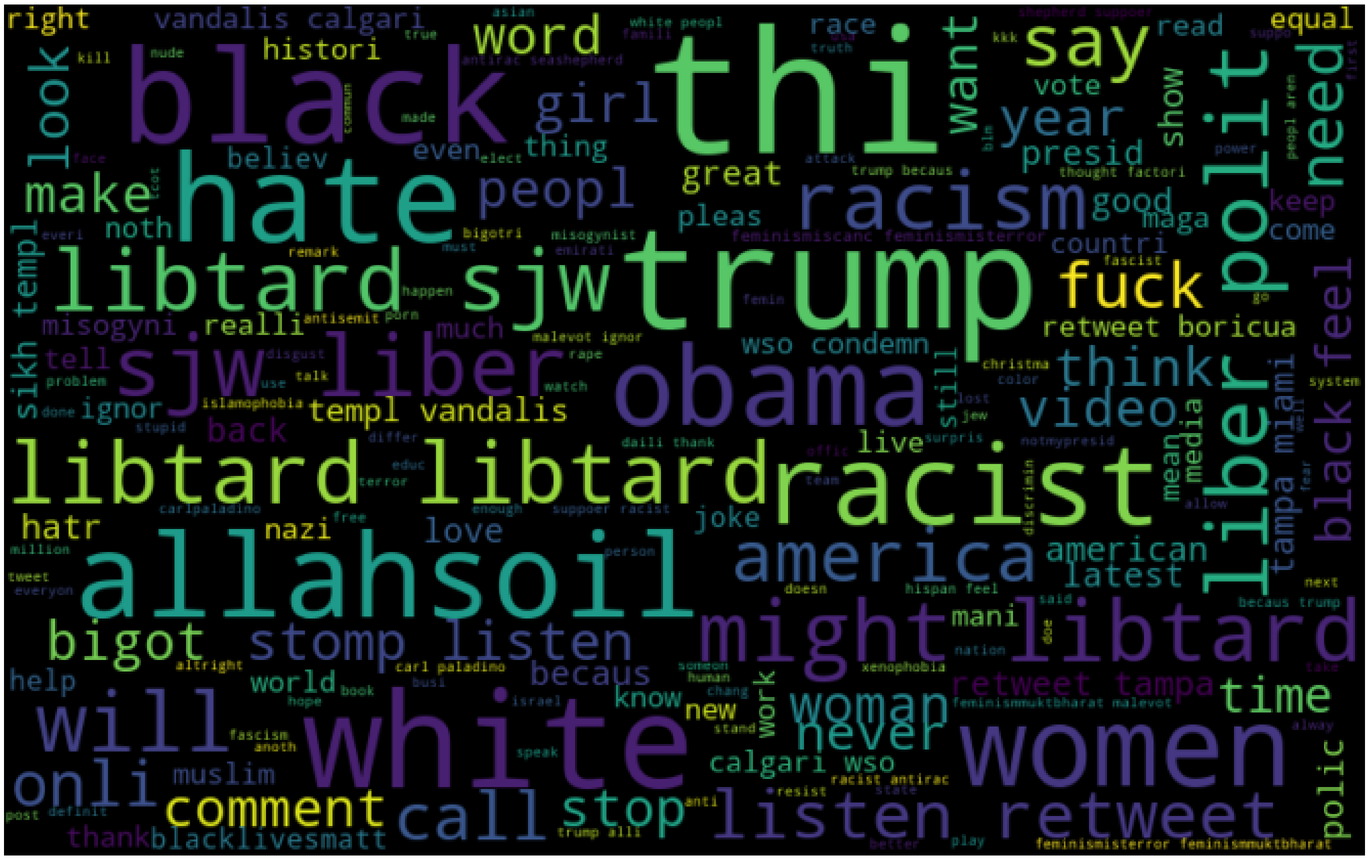


In [17]:
```
# frequent words visualization for -ve
all_words = " ".join([sentence for sentence in df['clean_tweet'][df['label']==1]])

wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).generat

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

Loading [MathJax]/extensions/Safe.js



In [18]:
```
# extract the hashtag
def hashtag_extract(tweets):
    hashtags = []
    # loop words in the tweet
    for tweet in tweets:
        ht = re.findall(r"#(\w+)", tweet)
        hashtags.append(ht)
    return hashtags
```

In [19]:
```
# extract hashtags from non-racist/sexist tweets
ht_positive = hashtag_extract(df['clean_tweet'][df['label']==0])

# extract hashtags from racist/sexist tweets
ht_negative = hashtag_extract(df['clean_tweet'][df['label']==1])
```

In [21]:
```
ht_positive[:5]
```

Out[21]:
```
[['run'], ['lyft', 'disapoint', 'getthank'], [], ['model'], ['motiv']]
```

In [22]:
```
# unnest list
ht_positive = sum(ht_positive, [])
ht_negative = sum(ht_negative, [])
```

In [23]:
```
ht_positive[:5]
```

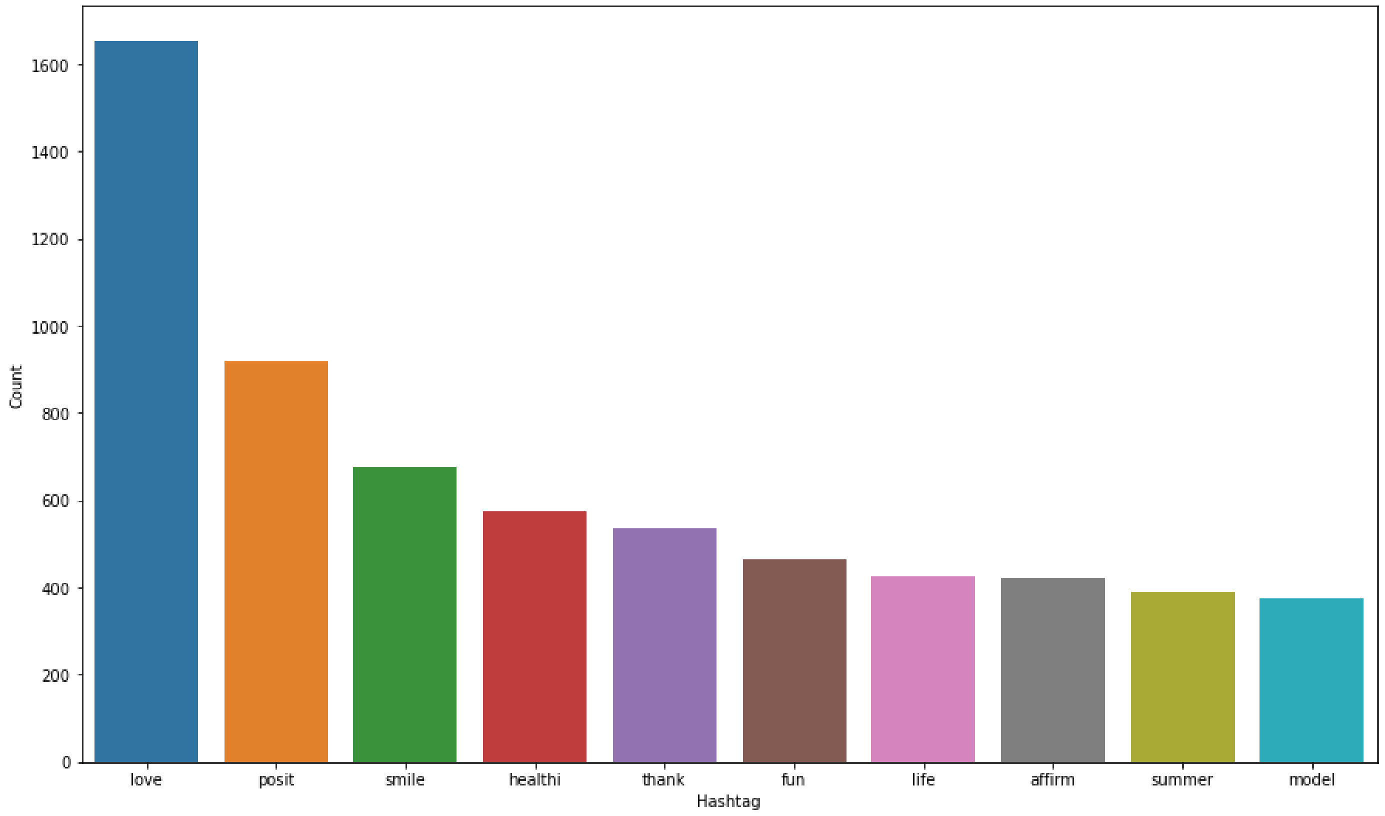
Out[23]:
```
['run', 'lyft', 'disapoint', 'getthank', 'model']
```

In [24]:
```
freq = nltk.FreqDist(ht_positive)
d = pd.DataFrame({'Hashtag': list(freq.keys()),
                  'Count': list(freq.values())})
d.head()
```

Out[24]:

	Hashtag	Count
0	run	72
1	lyft	2
2	disapoint	1
3	getthank	2
4	model	375

In [25]:
```
# select top 10 hashtags
d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hashtag', y='Count')
plt.show()
```



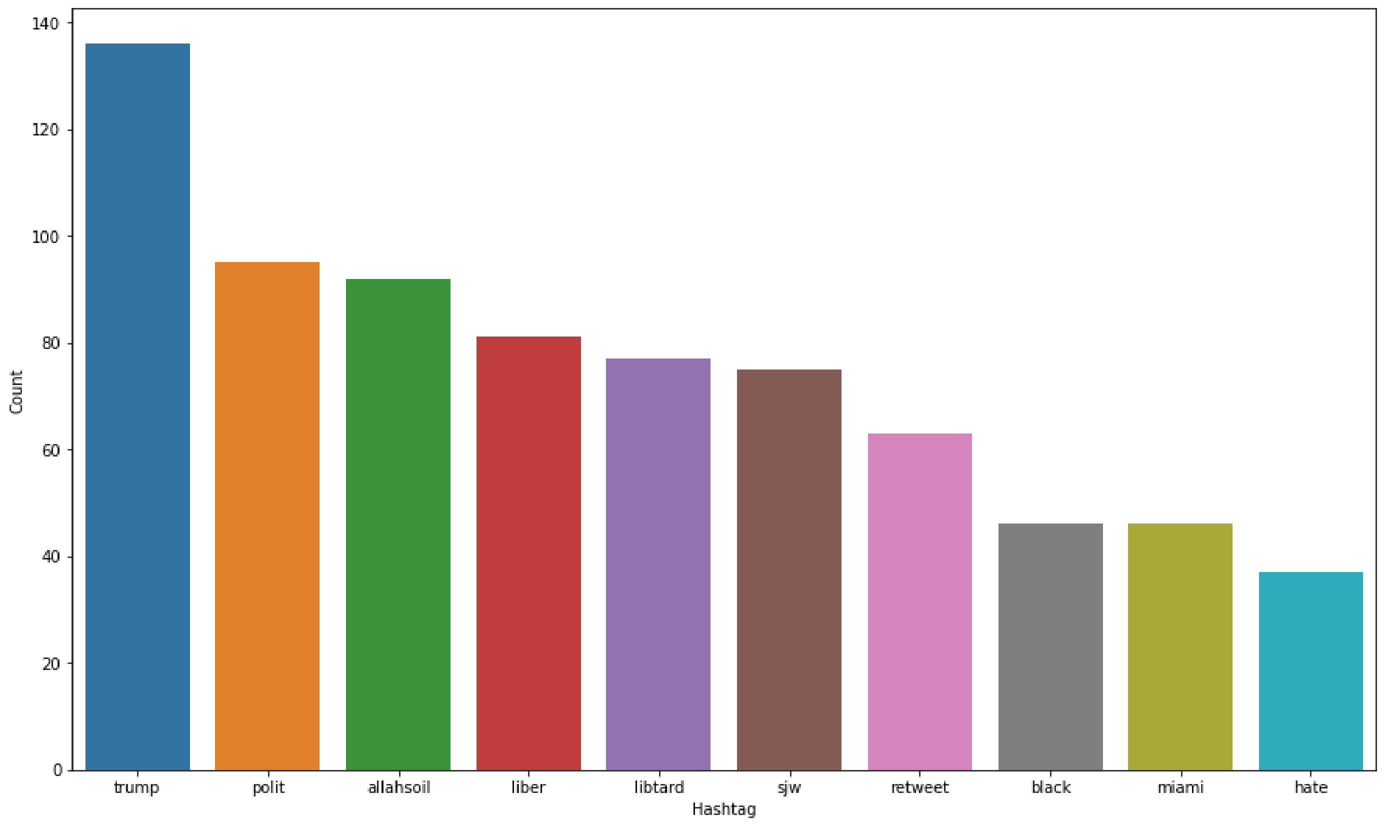
In [26]:
```
freq = nltk.FreqDist(ht_negative)
d = pd.DataFrame({'Hashtag': list(freq.keys()),
                  'Count': list(freq.values())})
d.head()
```

Out[26]:

	Hashtag	Count
0	cnn	10
1	michigan	2
2	tcot	14
3	australia	6
4	opkillingbay	5

In [27]:
```
# select top 10 hashtags
d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hashtag', y='Count')
```

Loading [MathJax]/extensions/Safe.js



Input Split

```
In [28]: # feature extraction
         from sklearn.feature_extraction.text import CountVectorizer
         bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='e
         bow = bow_vectorizer.fit_transform(df['clean_tweet'])
```

```
In [35]: # bow[0].toarray()
```

```
In [36]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(bow, df['label'], random_state=42, t
```

Model Training

```
In [41]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import f1_score, accuracy_score
```

```
In [38]: # training
         model = LogisticRegression()
         model.fit(x_train, y_train)

Out[38]: LogisticRegression()
```

```
In [39]: # testing
         pred = model.predict(x_test)
         f1_score(y_test, pred)

Out[39]: 0.49763033175355453
```

```
In [42]: accuracy_score(y_test,pred)
```

Loading [MathJax]/extensions/Safe.js

```
Out[42]: 0.94694030784632 71
```

```
In [43]: # use probability to get output
         pred_prob = model.predict_proba(x_test)
         pred = pred_prob[:, 1] >= 0.3
         pred = pred.astype(np.int)

         f1_score(y_test, pred)
Out[43]: 0.5545722713864307
```

```
In [44]: accuracy_score(y_test,pred)
Out[44]: 0.9433112251282693
```

```
In [47]: pred_prob[0][1] >= 0.3
Out[47]: False
```

```
In [ ]:
```

Loading [MathJax]/extensions/Safe.js