Twitter Hate Speech Analysis - Week 1

Colin Green, Sean Zhang

Problem Statement

Most individuals have encountered some form of hate speech or targeted harassment when participating on online social platforms such as social media, forums, or gaming. Unmoderated hate speech can have harmful consequences to its target and may cause anxiety, depression, or even suicide. Additionally, there are differences between hate speech and offensive language. Offensive language is merely speech that contains offensive words (such as quoting h*e or b*tch from song lyrics), while hate speech is targeted towards an individual or group, and is intended to be derogatory or insulting.

Moderation through a simple rule-based approach can help identify speech with offensive language - for example, checking if speech contains strings that match with a list of banned words. However, a rule-based approach would have difficulty differentiating between offensive language and hate speech, as they both employ offensive words. Previous research was able to accurately classify either hate speech or offensive language from 'clean' text, with a 91% and 95% accuracy respectively, but had low accuracy when differentiating between hate speech and offensive language from one another (61%). We plan to increase the accuracy in successfully identifying instances of hate speech from offensive language.

Disclaimer

This notebook contains uncensored offensive language for the purposes of data exploration and visualization.

Data Cleaning

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import seaborn as sns
pd.options.display.width = 500
import nltk
import string
import re
from nltk.stem.porter import *
```

In [2]:
```
df = pd.read_csv('labeled_data.csv')
df = df[df['class'] != 2].iloc[:,1:]
```

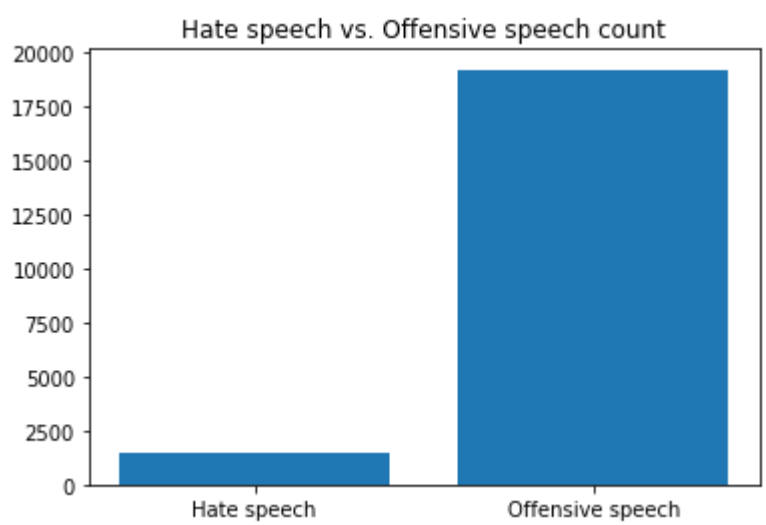
In [3]:
```
df.head()
```

Out[3]:

	count	hate_speech	offensive_language	neither	class	tweet
1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	0	2	1	1	!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	6	0	6	0	1	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
5	3	1	2	0	1	!!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just...

Data Exploration

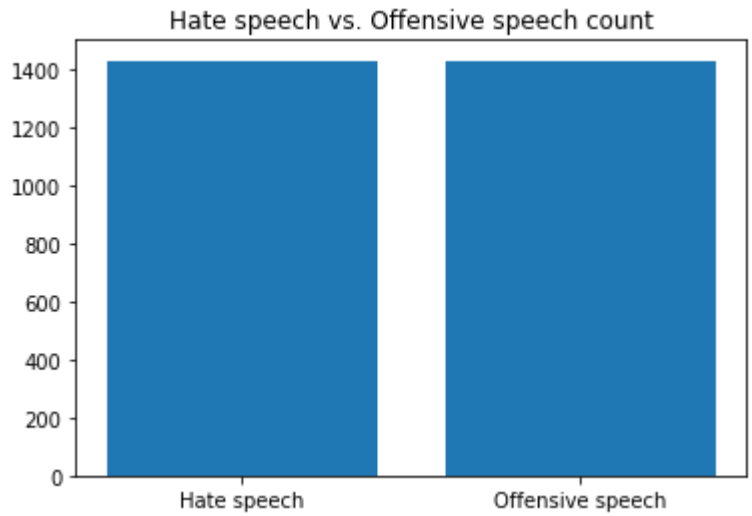
In [4]:
```
height = (len(df[df['class']==0]), len(df[df['class']==1]))
bars = ('Hate speech', 'Offensive speech')
plt.bar(bars, height)
plt.title('Hate speech vs. Offensive speech count')
plt.show()
lenhatespeech = len(df[df['class']==0])
lenoffspeech = len(df[df['class']==1])

print('Hate speech count: ', lenhatespeech, '(', round(100*lenhatespeech/(len
print('Offensive speech count: ', lenoffspeech, '(', round(100*lenoffspeech/(
```




```
Hate speech count:  1430 ( 6.94 %)
Offensive speech count:  19190 ( 93.06 %)
```

In [5]:
```
#Balance the dataset
np.random.seed(123)
remove_n = lenoffspeech - lenhatespeech
drop_indices = np.random.choice(df[df['class']==1].index, remove_n, replace=F
df_bal = df.drop(drop_indices)
```

In [6]:
```
height = (len(df_bal[df_bal['class']==0]), len(df_bal[df_bal['class']==1]))
bars = ('Hate speech', 'Offensive speech')
plt.bar(bars, height)
plt.title('Hate speech vs. Offensive speech count')
plt.show()
lenhatespeech = len(df_bal[df_bal['class']==0])
lenoffspeech = len(df_bal[df_bal['class']==1])

print('Hate speech count: ', lenhatespeech, '(', round(100*lenhatespeech/(len
print('Offensive speech count: ', lenoffspeech, '(', round(100*lenoffspeech/(
```




```
Hate speech count:  1430 ( 50.0 %)
Offensive speech count:  1430 ( 50.0 %)
```

In [7]:
```
#Data Preprocessing
import nltk
stopwords = nltk.corpus.stopwords.words('english')
other_exclusions = ["#ff", "ff", "rt"]
stopwords.extend(other_exclusions)
stemmer = PorterStemmer()


def preprocess(text_string):
    """
    Accepts a text string and replaces:
    1) urls with URLHERE
    2) lots of whitespace with one instance
    3) mentions with MENTIONHERE

    This allows us to get standardized counts of urls and mentions
    Without caring about specific people mentioned
    """
    space_pattern = '\s+'
    giant_url_regex = ('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|'
        '[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')
    mention_regex = '@[\w\-]+'
    hashtag_regex = '#[\w\-]+'
    parsed_text = re.sub(space_pattern, ' ', text_string)
    parsed_text = re.sub(giant_url_regex, 'URLHERE', parsed_text)
    parsed_text = re.sub(mention_regex, 'MENTIONHERE', parsed_text)
    parsed_text = re.sub(hashtag_regex, 'HASHTAGHERE', parsed_text)
    parsed_text = parsed_text.lower().strip()
    return parsed_text
```

In [8]:
```
df_bal['tweet_clean'] = ''
for i, row in df_bal.iterrows():
    df_bal.at[i, 'tweet_clean'] = preprocess(row.tweet)
```

In [9]:
```
df_bal['tweet_clean'].head()
```

Out[9]:
```
17                            " bitch who do you love "
23    " fuck no that bitch dont even suck dick " &ha...
38    " lames crying over hoes thats tears of a clown "
59    "..all i wanna do is get money and fuck model ...
62    "mentionhere: females think dating a pussy is ...
Name: tweet_clean, dtype: object
```

In [10]:
```
import spacy
nlp = spacy.load('en_core_web_sm')
```

In [11]:

```
for i, row in df_bal.iterrows():
    doc = nlp(str(row['tweet_clean']))
    sym = []
    nouns = []
    verbs = []
    lemmas = []

    for token in doc:
        lemmas.append(token.lemma_)
        if token.pos == 'SYM':
            sym.append(token.lemma_)
        if token.pos_ == 'NOUN' or token.pos_ == 'PROPN':
            nouns.append(token.lemma_)
        if token.pos_ == 'VERB':
            verbs.append(token.lemma_)

    df_bal.at[i, 'tweet_lemma'] = ' '.join(lemmas)
    df_bal.at[i, 'tweet_nouns'] = ' '.join(nouns)
    df_bal.at[i, 'tweet_sym'] = ' '.join(sym)
    df_bal.at[i, 'tweet_verbs'] = ' '.join(verbs)
    df_bal.at[i, 'tweet_nv'] = ' '.join(nouns + verbs)
    df_bal.at[i, 'num_tokens'] = len(lemmas)
```

In [12]:

```
df_bal.head()
```

Out[12]:

tweet	tweet_clean	tweet_lemma	tweet_nouns	tweet_sym	twee
ch who do you love "	" bitch who do you love "	" bitch who do -PRON- love "	bitch		
k no that bitch en suck dick " &#1...	" fuck no that bitch dont even suck dick " &ha...	" fuck no that bitch do not even suck dick " &...	bitch dick hashtaghere;&hashtaghere;&hashtaghe...		s
es crying over hats tears of a clown "	" lames crying over hoes thats tears of a clown "	" lame cry over hoe that s tear of a clown "	lame hoe tear clown		
anna do is get nd fuck model ...	"..all i wanna do is get money and fuck model ...	" .. all i wanna do be get money and fuck mode...	wanna money fuck model bitch russell simmon		
ZLEINDACUT: think dating a pussy...	"mentionhere: females think dating a pussy is ...	" mentionhere : female think date a pussy be c...	mentionhere female pussy stuff pussy		think

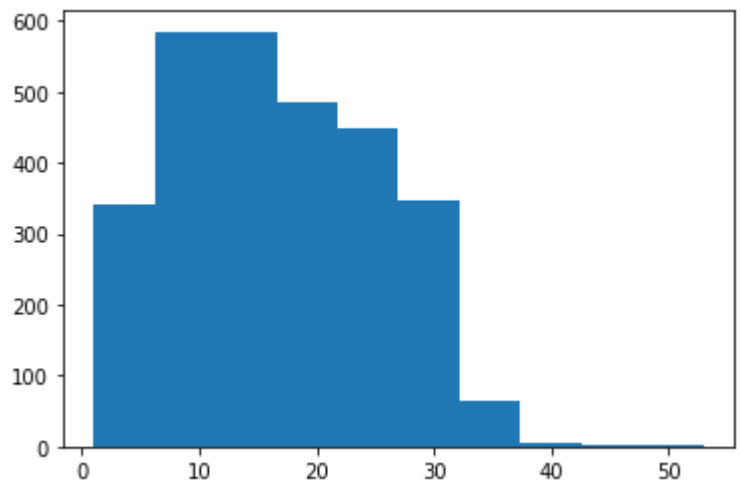
In [13]:
```
#Saving this to SQL database
import sqlite3
con = sqlite3.connect('twitter_hate.db')

drop = '''drop table tweets_nlp'''
with sqlite3.connect('twitter_hate.db') as con:
    con.execute(drop)
df_bal.to_sql('tweets_nlp', con)
sql = """
SELECT * FROM tweets_nlp
"""
with sqlite3.connect('twitter_hate.db') as con:
    df_bal = pd.read_sql_query(sql, con)
```

In [14]:
```
print('Data types:\n', df_bal.dtypes)
print('Shape: ', df_bal.shape)
print('Count:\n', df_bal.count())
```

```
Data types:
 index                   int64
count                   int64
hate_speech             int64
offensive_language      int64
neither                 int64
class                   int64
tweet                  object
tweet_clean            object
tweet_lemma            object
tweet_nouns            object
tweet_sym              object
tweet_verbs            object
tweet_nv               object
num_tokens            float64
dtype: object
Shape:  (2860, 14)
Count:
 index                 2860
count                 2860
hate_speech           2860
offensive_language    2860
neither               2860
class                 2860
tweet                 2860
tweet_clean           2860
tweet_lemma           2860
tweet_nouns           2860
tweet_sym             2860
tweet_verbs           2860
tweet_nv              2860
num_tokens            2860
dtype: int64
```

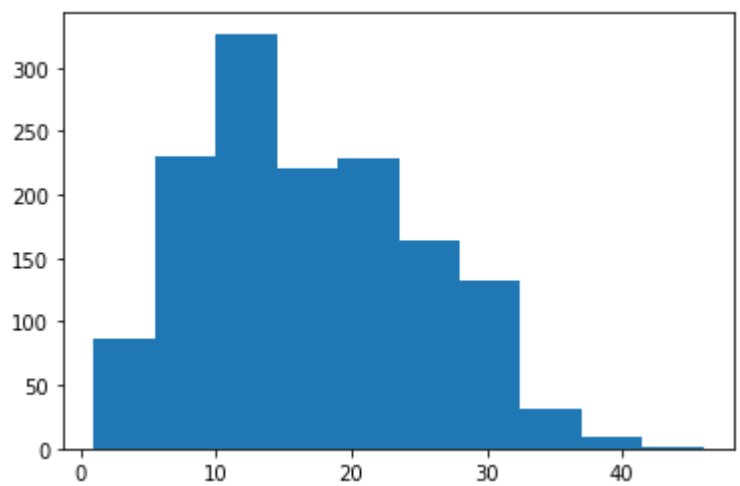
In [15]:
```
plt.hist(df_bal['num_tokens'])
print('Average number of tokens per tweet: ', round(df_bal['num_tokens'].mear
```

```
Average number of tokens per tweet:  16.62
```



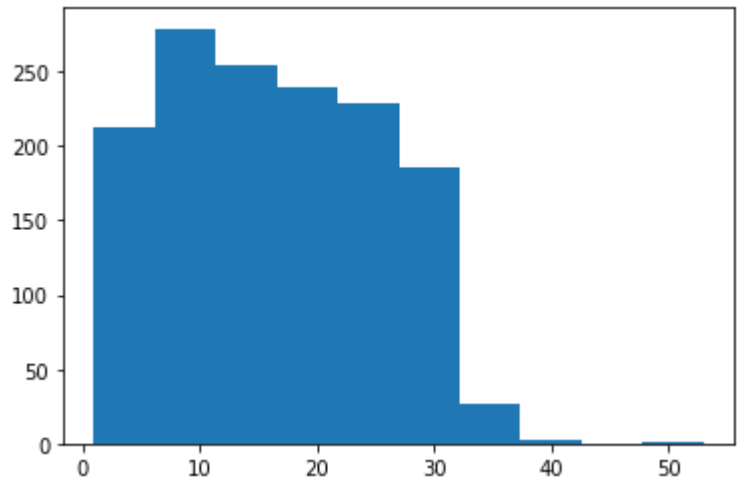
In [16]:
```
df_offensive = df_bal[df_bal['class']==1]
plt.hist(df_offensive['num_tokens'])
print('Average number of tokens per OFFENSIVE tweet: ', round(df_offensive['r
```

```
Average number of tokens per OFFENSIVE tweet:  16.8
```



In [17]:
```
df_hate = df_bal[df_bal['class']==0]
plt.hist(df_hate['num_tokens'])
print('Average number of tokens per HATE tweet: ', round(df_hate['num_tokens'

# Hate tweets have slightly fewer tokens on average and a greater proportion
# This is in line with a presentation at the TMLS conference from Washington
```

```
Average number of tokens per HATE tweet:  16.45
```



In [18]:
```
#here we look at just the noun adjective and verbs
def my_tokenizer(text):
    return text.split() if text != None else []
tokens = df_bal.tweet_nv.map(my_tokenizer).sum()
tokens[:20]
```

Out[18]:
```
['bitch',
 'love',
 'bitch',
 'dick',
 'hashtaghere;&hashtaghere;&hashtaghere',
 'kermit',
 'video',
 'suck',
 'bout',
 'fuck',
 'lame',
 'hoe',
 'tear',
 'clown',
 'cry',
 's',
 'wanna',
 'money',
 'fuck',
 'model']
```

In [19]:
```
from collections import Counter

counter = Counter(tokens)
counter.most_common(20)
```

Out[19]:
```
[('mentionhere', 2234),
 ('bitch', 1099),
 ('rt', 744),
 ('hoe', 383),
 ('faggot', 252),
 ('get', 242),
 ('nigga', 236),
 ('fuck', 220),
 ('be', 218),
 ('ass', 191),
 ('go', 181),
 ('can', 180),
 ('nigger', 168),
 ('hashtaghere;mentionhere', 165),
 ('pussy', 144),
 ('trash', 139),
 ('shit', 135),
 ('u', 127),
 ('say', 124),
 ('fag', 123)]
```

In [20]:
```
from spacy.lang.en.stop_words import STOP_WORDS
stopwords.append(STOP_WORDS)

def remove_stopwords(tokens):
    return [t for t in tokens if t not in stopwords]

counter = Counter(remove_stopwords(tokens))
counter.most_common(20)
```

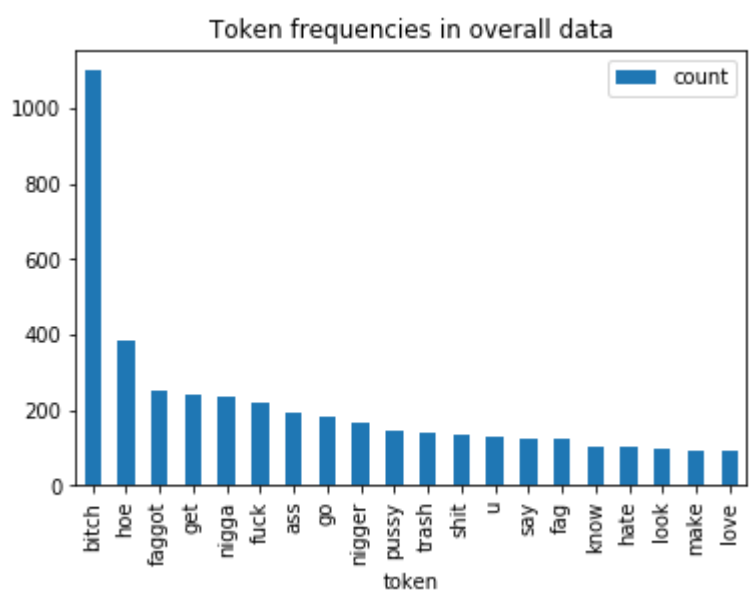
Out[20]:
```
[('mentionhere', 2234),
 ('bitch', 1099),
 ('hoe', 383),
 ('faggot', 252),
 ('get', 242),
 ('nigga', 236),
 ('fuck', 220),
 ('ass', 191),
 ('go', 181),
 ('nigger', 168),
 ('hashtaghere;mentionhere', 165),
 ('pussy', 144),
 ('trash', 139),
 ('shit', 135),
 ('u', 127),
 ('say', 124),
 ('fag', 123),
 ('know', 102),
 ('hate', 101),
 ('look', 97)]
```

In [21]:
```
#MENTIONHERE, RT, HASHTAGHERE, URLHERE, HASTAGHERE;MENTIONHERE are not actual
#amp likely refers to ampersand
#Let's remove these:

ignore_counter = ['mentionhere', 'hashtaghere', 'urlhere', 'hashtaghere;menti
for word in list(counter):
    if word in ignore_counter:
        del counter[word]
counter.most_common(20)
```

Out[21]:
```
[('bitch', 1099),
 ('hoe', 383),
 ('faggot', 252),
 ('get', 242),
 ('nigga', 236),
 ('fuck', 220),
 ('ass', 191),
 ('go', 181),
 ('nigger', 168),
 ('pussy', 144),
 ('trash', 139),
 ('shit', 135),
 ('u', 127),
 ('say', 124),
 ('fag', 123),
 ('know', 102),
 ('hate', 101),
 ('look', 97),
 ('make', 94),
 ('love', 93)]
```

In [22]:
```
#convert list to bargraph
freq_df = pd.DataFrame.from_records(counter.most_common(20), columns = ['toke
freq_df.plot(kind = 'bar', x = 'token', title = 'Token frequencies in overall
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c7d5afd588>
```



In [23]:
```
from wordcloud import WordCloud
from matplotlib import pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'svg'
plt.rcParams['figure.dpi'] = 100

def wordcloud(counter):
    wc = WordCloud(width = 1200, height = 800, background_color = 'white', ma
    wc.generate_from_frequencies(counter)

    fig = plt.figure(figsize=(6,4))
    plt.imshow(wc, interpolation = 'bilinear')
    plt.axis('off')
    plt.show()
```

In [24]:
```
wordcloud(counter)
```

In [25]:
```
#Now see if there are any differences between hate speech and offensive speec
tokens_offensive = df_offensive.tweet_nv.map(my_tokenizer).sum()
counter_offensive = Counter(remove_stopwords(tokens_offensive))
for word in list(counter_offensive):
    if word in ignore_counter:
        del counter_offensive[word]
counter_offensive.most_common(20)
```

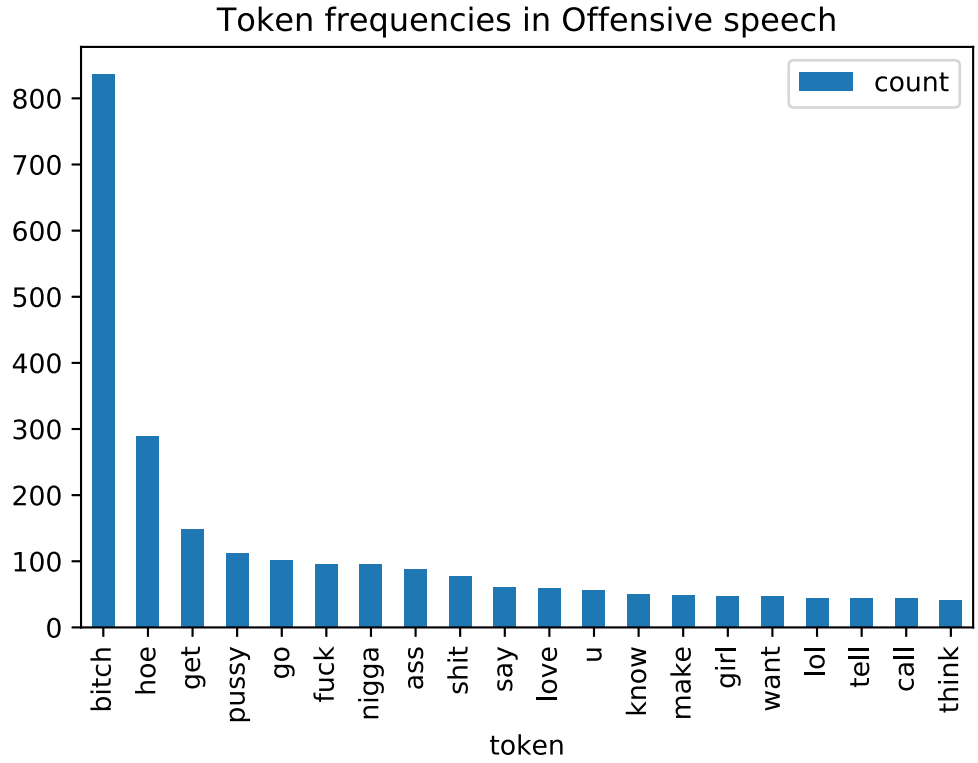
Out[25]:
```
[('bitch', 836),
 ('hoe', 289),
 ('get', 149),
 ('pussy', 113),
 ('go', 102),
 ('fuck', 96),
 ('nigga', 96),
 ('ass', 88),
 ('shit', 77),
 ('say', 61),
 ('love', 59),
 ('u', 56),
 ('know', 51),
 ('make', 49),
 ('girl', 48),
 ('want', 47),
 ('lol', 44),
 ('tell', 44),
 ('call', 44),
 ('think', 42)]
```

In [26]:
```
tokens_hate = df_hate.tweet_nv.map(my_tokenizer).sum()
counter_hate = Counter(remove_stopwords(tokens_hate))
for word in list(counter_hate):
    if word in ignore_counter:
        del counter_hate[word]
counter_hate.most_common(20)
```

Out[26]:
```
[('bitch', 263),
 ('faggot', 233),
 ('nigger', 161),
 ('nigga', 140),
 ('fuck', 124),
 ('trash', 112),
 ('fag', 107),
 ('ass', 103),
 ('hoe', 94),
 ('get', 93),
 ('go', 79),
 ('u', 71),
 ('hate', 67),
 ('say', 63),
 ('people', 59),
 ('shit', 58),
 ('niggas', 58),
 ('look', 56),
 ('know', 51),
 ('make', 45)]
```

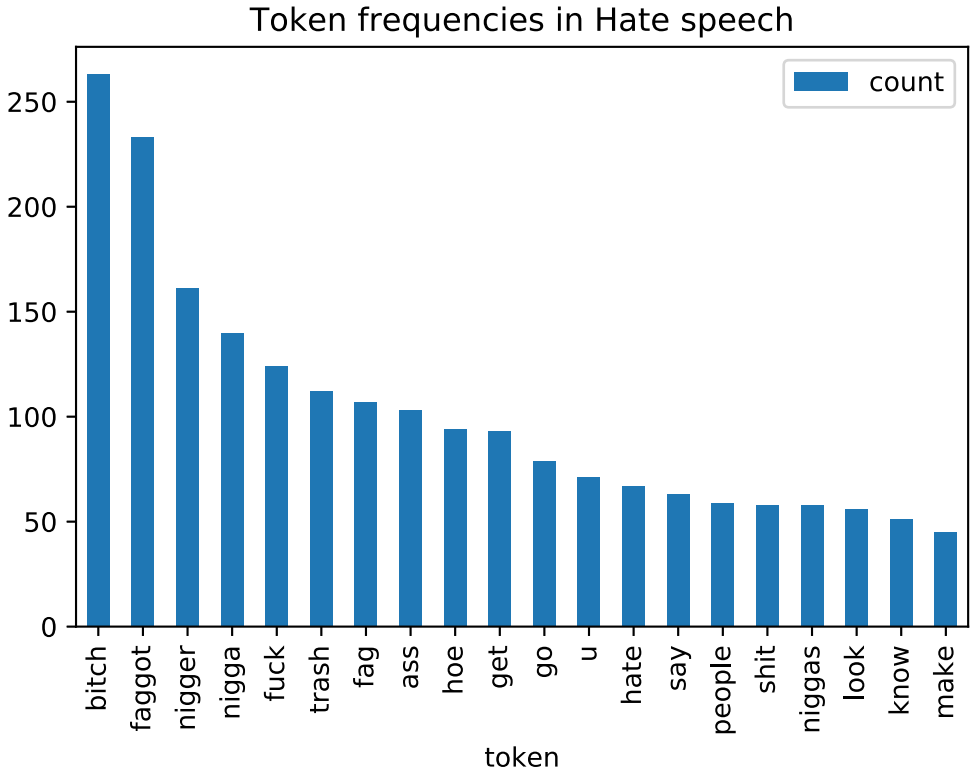
In [27]:
```
freq_df = pd.DataFrame.from_records(counter_offensive.most_common(20), column
freq_df.plot(kind = 'bar', x = 'token', title = 'Token frequencies in Offensi
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c7d6461088>
```



In [28]:
```
freq_df = pd.DataFrame.from_records(counter_hate.most_common(20), columns = [
freq_df.plot(kind = 'bar', x = 'token', title = 'Token frequencies in Hate sp
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1c7d64f7288>



In [29]:
```
#Word Cloud of offensive speech
wordcloud(counter_offensive)
```

In [30]:
```
#Word Cloud of hate speech
wordcloud(counter_hate)
```

An exploratory analysis shows that hate speech tends to be less complex (fewer tokens) and uses stronger homophobic and racial slurs (f*g, n-word ending with 'hard R' sound).