0203_hw_exemplar

January 14, 2020

In [2]:
```
# Import packages for data cleaning,
# viz and some sentiment analysis for fun :)

import pandas as pd
import nltk
nltk.download('vader_lexicon')
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\jerem\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [3]:
```
# Data files to be used. Political Twitter Data.
#sourceDir = ""
tweetsFile = "ExtractedTweets.csv"
twitterHandles = "TwitterHandles.csv"
```

In [4]:
```
justTweetsDf = pd.read_csv(tweetsFile)
justHandlesDf = pd.read_csv(twitterHandles)
```

In [5]:
```
# Check to confirm data loaded correctly
justTweetsDf.head()
```

Out[5]:

	Party	Handle	Tweet
0	Democrat	RepDarrenSoto	Today, Senate Dems vote to #SaveTheInternet. P...
1	Democrat	RepDarrenSoto	RT @WinterHavenSun: Winter Haven resident / Al...
2	Democrat	RepDarrenSoto	RT @NBCLatino: .@RepDarrenSoto noted that Hurr...
3	Democrat	RepDarrenSoto	RT @NALCABPolicy: Meeting with @RepDarrenSoto ...
4	Democrat	RepDarrenSoto	RT @Vegalteno: Hurricane season starts on June...

In [6]:
```
# Begin to investigate data. Find num of Dem and Rep

print(len(justTweetsDf[justTweetsDf['Party']=='Democrat']))
print()
print(len(justTweetsDf[justTweetsDf['Party']=='Republican']))
print()
print(len(justTweetsDf))
```

```
42068

44392

86460
```

```
In [7]: print(justHandlesDf.shape)
        justHandlesDf = justHandlesDf.drop_duplicates(subset='Handle')
        print(justHandlesDf.shape)
```

```
(48, 4)
(40, 4)
```

```
In [8]: justHandlesDf.head()
```

Out[8]:

	Party	Name	Handle	AvatarURL
0	Democrat	US Rep. Darren Soto	RepDarrenSoto	https://pbs.twimg.com/profile_images/824454906...
1	Democrat	Rep. Jacky Rosen	RepJackyRosen	https://pbs.twimg.com/profile_images/837772241...
2	Democrat	US Rep. Al Lawson Jr	RepAlLawsonJr	https://pbs.twimg.com/profile_images/818493713...
3	Democrat	Adriano Espaillat	RepEspaillat	https://pbs.twimg.com/profile_images/827580972...
8	Democrat	Rep. Blunt Rochester	RepBRochester	https://pbs.twimg.com/profile_images/912673706...

```
In [9]: tweetsDf = justTweetsDf.merge(copy=True, right=justHandlesDf, on='Handle', how='left')
        tweetsDf = tweetsDf.drop(columns=['Party_y','AvatarURL'])
        tweetsDf = tweetsDf.rename(columns={'Party_x':'Party'})
```

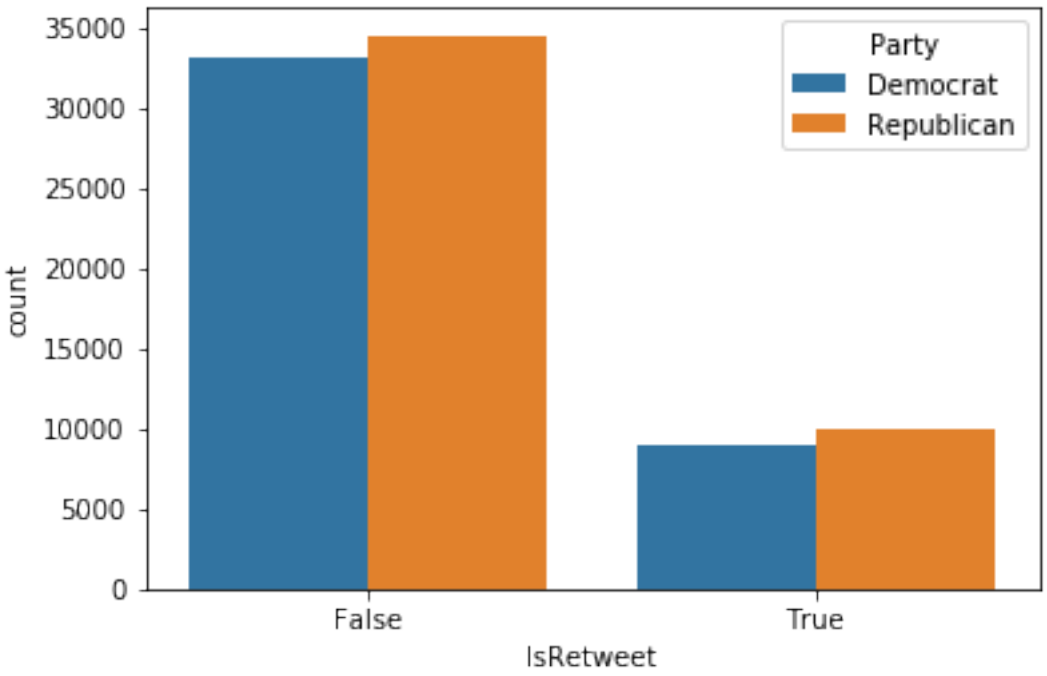
```
In [10]: len(tweetsDf['Handle'].unique())
```

```
Out[10]: 433
```

```
In [11]: # Clean some of the text data. EG get rid of "RT"

         tweetsDf['IsRetweet'] = tweetsDf['Tweet'].str.startswith('RT')
         tweetsDf['Tweet'] = tweetsDf['Tweet'].str.replace('RT ', '').str.strip()
```

```
In [12]: # More EDA!! Categorize tweets; were they rts?

         sns.countplot(data=tweetsDf, x='IsRetweet', hue='Party')
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1725fc44eb8>
```




```
In [13]: # Quartile Analysis on wordcount

         tweetsDf['WordCount'] = tweetsDf['Tweet'].str.count("\S\s+\S")+1
         tweetsDf['WordCount'].describe()
```

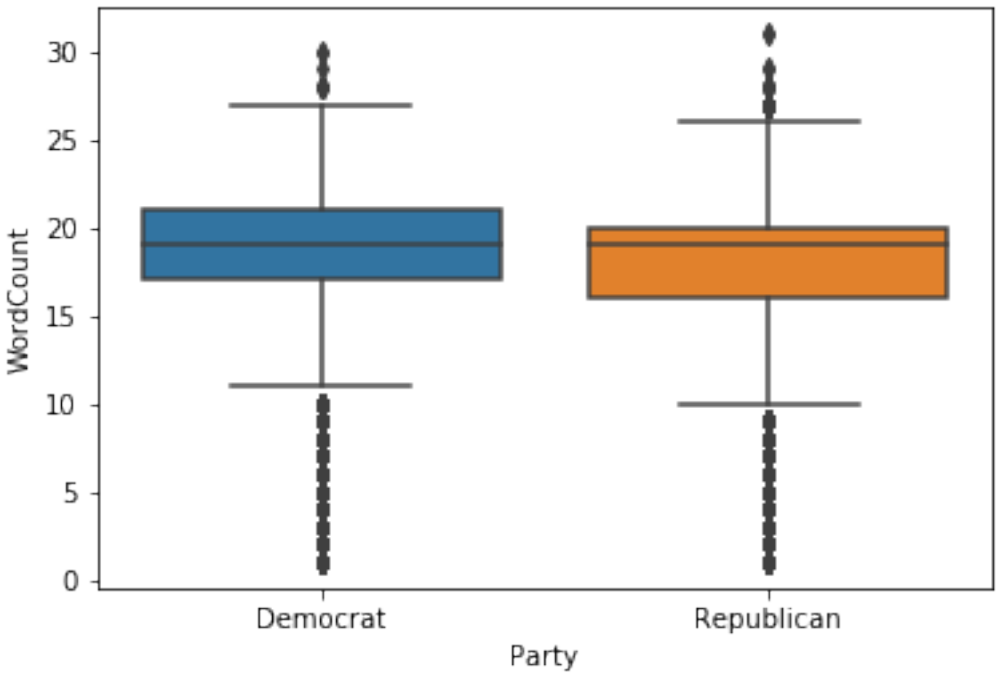
```
Out[13]: count    86460.000000
         mean        17.823225
         std          4.253023
         min          1.000000
         25%         16.000000
         50%         19.000000
         75%         21.000000
         max         31.000000
         Name: WordCount, dtype: float64
```

```
In [14]: # More EDA!! Peak at data and compute mean word counts

         print(tweetsDf['Tweet'][0])
         print(tweetsDf['WordCount'][0])

         print(tweetsDf[tweetsDf['Party']=='Democrat']['WordCount'].mean())
         print(tweetsDf[tweetsDf['Party']=='Republican']['WordCount'].mean())

         print(len(tweetsDf[tweetsDf['Party']=='Democrat']))
         print(len(tweetsDf[tweetsDf['Party']=='Republican']))
```

```
Today, Senate Dems vote to #SaveTheInternet. Proud to support similar #NetNeutrality legislatio
17
18.045878102120376
17.612227428365472
42068
44392
```

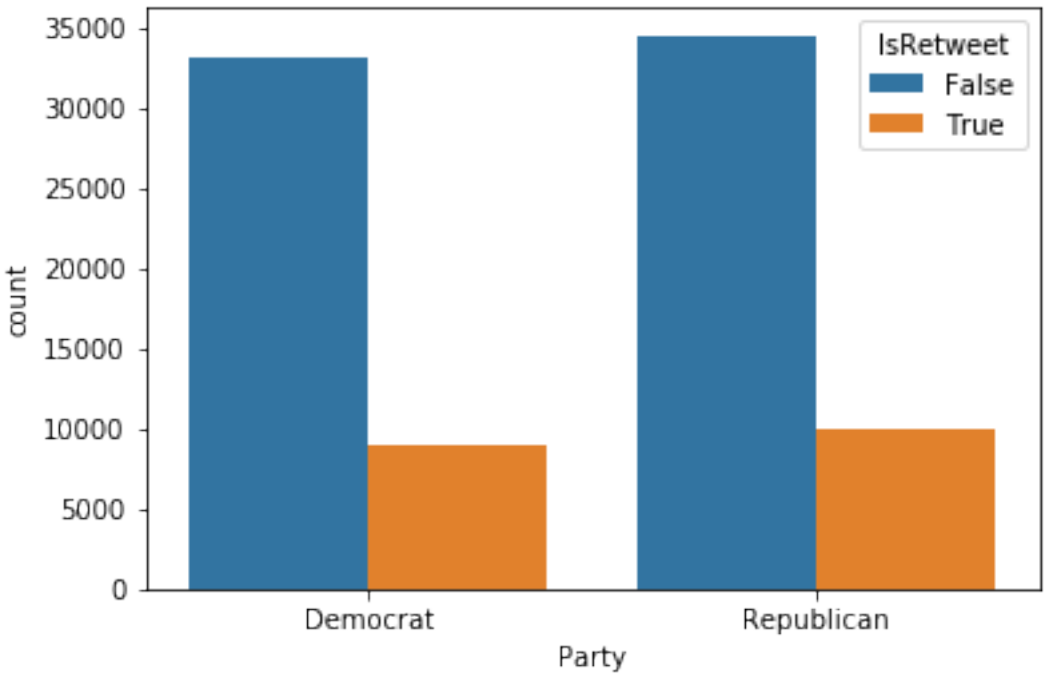
In [15]:
```
#print(tweetsDf['WordCount'].describe())
sns.boxplot(data=tweetsDf, x='Party', y='WordCount')
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1725ff750b8>



In [16]:
```
sns.countplot(data=tweetsDf, hue='IsRetweet', x='Party')
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1725ff876a0>



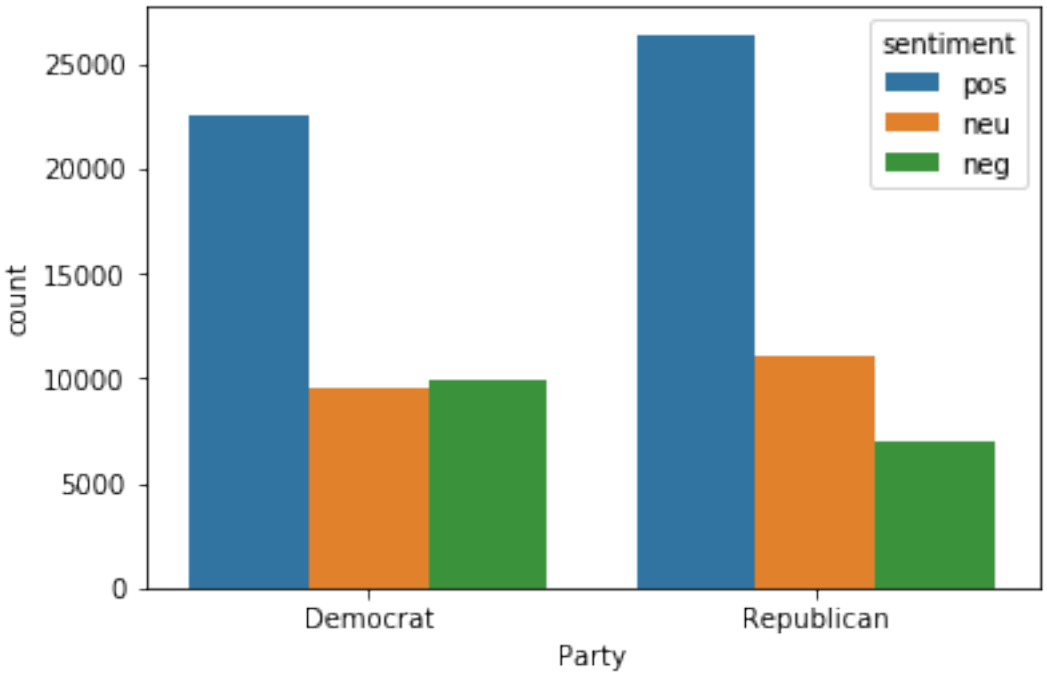
In [17]:
```
# After loading + cleaning the data, try some back end analysis for fun :)
# We will learn more about sentiment analysis and these packages later on
# This is simply a "teaser" intro

from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.sentiment import SentimentAnalyzer

sid = SentimentIntensityAnalyzer()
sentimentScoreAry = []
sentimentAry = []

# Use vader to assign sentiment to tweets.

for t in tweetsDf['Tweet']:
    ss = sid.polarity_scores(t)['compound']
    sentimentScoreAry.append(ss)
    if (ss > 0):
        sentimentAry.append('pos')
    elif (ss < 0):
        sentimentAry.append('neg')
    else:
        sentimentAry.append('neu')

sentimentScoreAry
```

```
        tweetsDf['sentiment_score'] = sentimentScoreAry
        tweetsDf['sentiment'] = sentimentAry
```

```
C:\Users\jerem\Anaconda3\lib\site-packages\nltk\twitter\__init__.py:20: UserWarning: The twytho
  warnings.warn("The twython library has not been installed. "
```

In [18]:
```
# Viz the sentiment

sns.countplot(data=tweetsDf, hue='sentiment', x='Party')
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1726003f4a8>
```

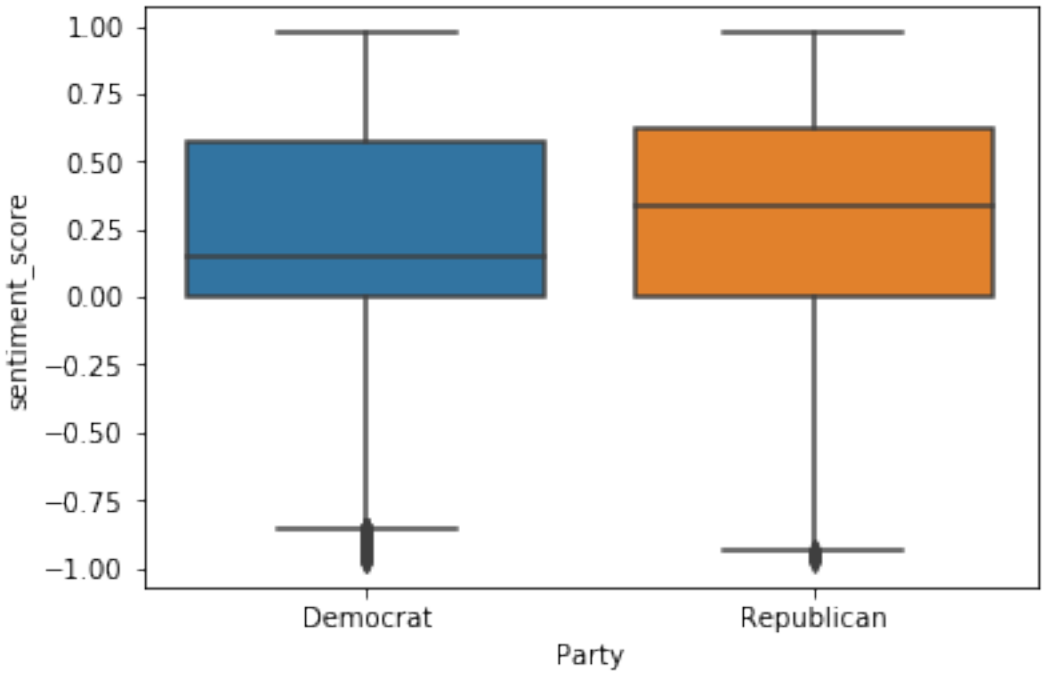


In [19]:
```
# Basic EDA with Sent Labels

tweetsDf['sentiment'].describe()
```

Out[19]:
```
count     86460
unique        3
top         pos
freq      48936
Name: sentiment, dtype: object
```

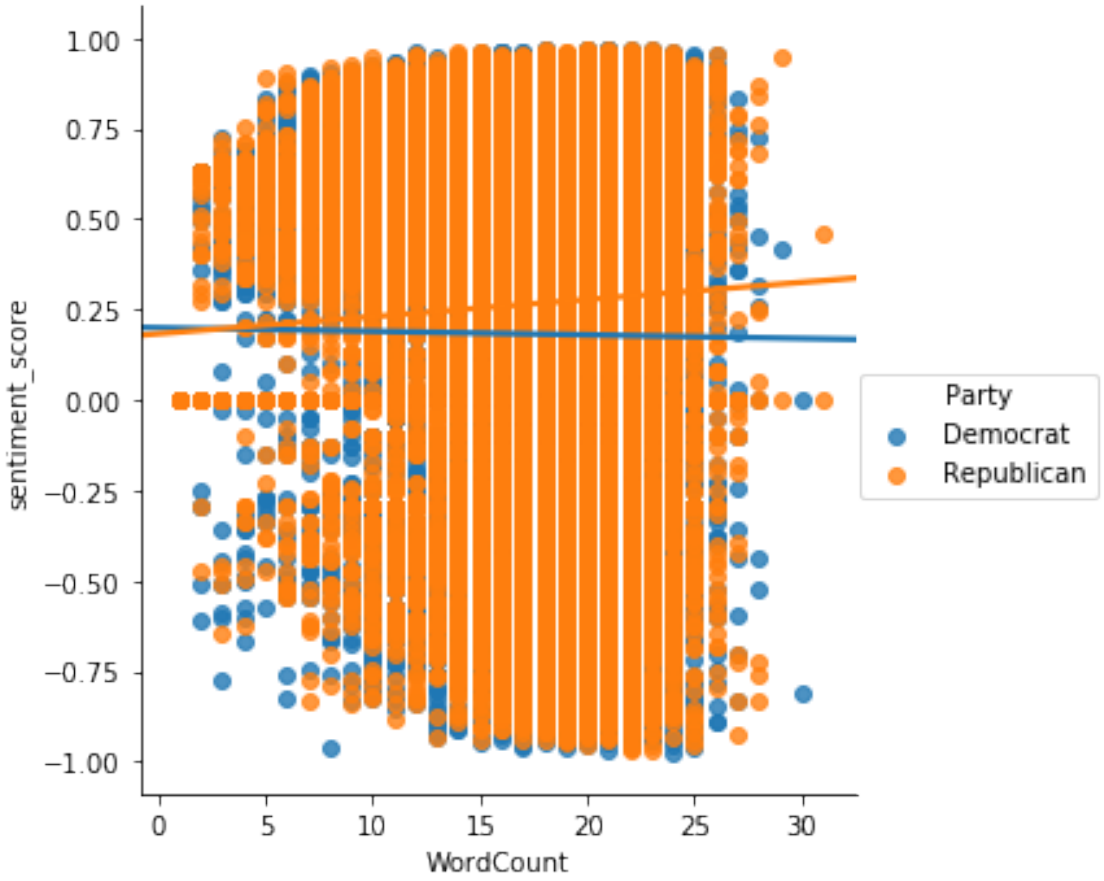
In [20]:
```
#print(tweetsDf['WordCount'].describe())
sns.boxplot(data=tweetsDf, x='Party', y='sentiment_score')
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x172601ea198>



In [23]:
```
# Is there a relationship between word count and sentiment???

sns.lmplot(data=tweetsDf, x='WordCount', y='sentiment_score', hue='Party', ci=False)
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x17262d9b1d0>



In [24]:
```
# Peak at small tweets, is there a pattern wrt sentiment?

tweetsDf[tweetsDf['WordCount'] < 5]
```

Out[24]:

	Party	Handle	\
52	Democrat	RepDarrenSoto	
118	Democrat	RepDarrenSoto	
254	Democrat	RepJackyRosen	
355	Democrat	RepJackyRosen	
423	Democrat	RepAlLawsonJr	
632	Democrat	RepEspaillat	
633	Democrat	RepEspaillat	
857	Democrat	RepBRochester	
935	Democrat	RepBRochester	
1023	Democrat	RepBarragan	
1026	Democrat	RepBarragan	
1204	Democrat	RepTomSuozzi	
1210	Democrat	RepTomSuozzi	
1215	Democrat	RepTomSuozzi	

```
1217     Democrat     RepTomSuozzi
1218     Democrat     RepTomSuozzi
1219     Democrat     RepTomSuozzi
1220     Democrat     RepTomSuozzi
1221     Democrat     RepTomSuozzi
1235     Democrat     RepTomSuozzi
1339     Democrat     RepTomSuozzi
1607     Democrat        RepKihuen
1681     Democrat        RepKihuen
1743     Democrat        RepKihuen
1795     Democrat        RepKihuen
1811     Democrat      RepMcEachin
1938     Democrat      RepMcEachin
1957     Democrat      RepMcEachin
2202     Democrat  RepCharlieCrist
2275     Democrat  RepCharlieCrist
...           ...              ...
83642  Republican       farenthold
83825  Republican       ToddRokita
84060  Republican      EdWorkforce
84122  Republican      EdWorkforce
84188  Republican      EdWorkforce
84331  Republican     virginiafoxx
84354  Republican     virginiafoxx
84429  Republican     virginiafoxx
84445  Republican     virginiafoxx
84526  Republican     davereichert
84681  Republican   LamarSmithTX21
84736  Republican   LamarSmithTX21
84772  Republican   LamarSmithTX21
84787  Republican   LamarSmithTX21
84812  Republican   LamarSmithTX21
85051  Republican  michaelcburgess
85063  Republican       RepShimkus
85115  Republican       RepShimkus
85149  Republican       RepShimkus
85224  Republican       RepShimkus
85236  Republican       RepShimkus
85509  Republican       RobWittman
85575  Republican       RobWittman
85608  Republican       RobWittman
85612  Republican       RobWittman
85707  Republican      RosLehtinen
85836  Republican      RosLehtinen
85902  Republican  WaysandMeansGOP
85904  Republican  WaysandMeansGOP
86069  Republican        GOPpolicy
```

	Tweet
52	https://t.co/RPRVK1GNtX
118	https://t.co/nVnVrFyJUM
254	Incredible! #GirlsWhoCode https://t.co/Ex1y0lq6qj
355	Welcome, Congressman @ConorLambPA! https://t.c...
423	https://t.co/HpOF0FZv2v
632	https://t.co/Mq5NCdMICD
633	https://t.co/K4vbv41DoY
857	https://t.co/cPKm1aZRzC
935	#GetCovered https://t.co/nwj5i9185l
1023	@RepRoKhanna Thanks!
1026	Congratulations! https://t.co/FWXNrJ4dE3
1204	@randimarshall @Amtrak Thank you!
1210	Thank you @SkyeStats https://t.co/opxeDXBOio
1215	Thank you, @COPUR_National
1217	Thank you @themishpacha https://t.co/NEt5XcKY5Z
1218	https://t.co/p4cfnOfztu https://t.co/3vRrxKFqUM
1219	Thank you, @longisland https://t.co/f4s2iHUGAo
1220	@Local338
1221	Great night! #UnionStrong https://t.co/4HmBmHh4oK
1235	@HuntTownHall
1339	Wishing everyone a very #Happy2018!
1607	VGK!!! #VegasBorn https://t.co/hDTfpc7hXP
1681	Congrats, @GoldenKnights!!! #VegasBorn https:/...
1743	Congratulations, @GoldenKnights! #VegasBorn ht...
1795	@HeyMeelahDee: Yasssssssssss #EqualityForAll h...
1811	Happy #MothersDay! https://t.co/OZ5P7dLBrg
1938	Outstanding News. https://t.co/r8oQTTckAB
1957	#EarthDay https://t.co/GubRgKOsej
2202	Congrats Pinellas grads! https://t.co/ct3FX0...
2275	Happy Easter! https://t.co/q8JiKSJqqN
...	...
83642	Happy #DoubleTenDay Taiwan! https://t.co/Me7Hh...
83825	@cathymcmorris Thank you, @cathymcmorris!
84060	Welcome! https://t.co/RP73hBVmDR
84122	@RepJoeWilson: Tune in! https://t.co/n1VFfDWhUz
84188	More https://t.co/Nw2K47ARmo
84331	Happy Birthday, @SpeakerRyan ! https://t.co/31...
84354	Happy Thanksgiving! https://t.co/yplcoiQLiV
84429	Happy Father's Day!
84445	https://t.co/6wfsun2xZD
84526	Congratulations @TahomaHigh! https://t.co/tIn1...
84681	Congrats @ReaganBandSA! https://t.co/6wFmemGk4K
84736	Happy Easter! https://t.co/1WoqRc4iRA
84772	@TxStHistAssoc: https://t.co/tViWMkx3mW
84787	Tomorrow morning -> https://t.co/KByOJHldYj
84812	@SpecNewsSA: https://t.co/E1AvTFJTZ5
85051	Thanks @EW_UNT #GMG https://t.co/8k52x2Bytk

```
85063  @CarlResists @AmeriCorps @SeniorCorps https://...
85115          Great game, Erik! https://t.co/onkvy7azGC
85149     https://t.co/LpUeTjgdcX https://t.co/Hsp...
85224  @MsCharlieJohnso Positive. https://t.co/WUIOvB...
85236                Fixed link: https://t.co/O7O3I5Z9Pi
85509    Happy #NationalParkWeek https://t.co/2GnYMOnkyh
85575             Full statement: https://t.co/xwBI2KnKUG
85608           Let's Go Hokies! https://t.co/O9q1WrfOwQ
85612           Fair assessment. https://t.co/u33xSFAGck
85707     Gracias @DLasAmericas! https://t.co/YUbIWoTqjf
85836     Gracias @DLasAmericas! https://t.co/cj2UNkvhNx
85902               @PeterRoskam: https://t.co/ETliUWGvJc
85904           #HappyMothersDay https://t.co/kDdN1JvRz8
86069      We will #NeverForget. https://t.co/097kB5vI5Q
```

	Name	IsRetweet	WordCount	sentiment_score	sentiment
52	US Rep. Darren Soto	False	1	0.0000	neu
118	US Rep. Darren Soto	False	1	0.0000	neu
254	Rep. Jacky Rosen	False	3	0.0000	neu
355	Rep. Jacky Rosen	False	4	0.5093	pos
423	US Rep. Al Lawson Jr	False	1	0.0000	neu
632	Adriano Espaillat	False	1	0.0000	neu
633	Adriano Espaillat	False	1	0.0000	neu
857	Rep. Blunt Rochester	False	1	0.0000	neu
935	Rep. Blunt Rochester	False	2	0.0000	neu
1023	Nanette D. Barragán	False	2	0.4926	pos
1026	Nanette D. Barragán	False	2	0.6360	pos
1204	Tom Suozzi	False	4	0.4199	pos
1210	Tom Suozzi	False	4	0.3612	pos
1215	Tom Suozzi	False	3	0.3612	pos
1217	Tom Suozzi	False	4	0.3612	pos
1218	Tom Suozzi	False	2	0.0000	neu
1219	Tom Suozzi	False	4	0.3612	pos
1220	Tom Suozzi	False	1	0.0000	neu
1221	Tom Suozzi	False	4	0.6588	pos
1235	Tom Suozzi	False	1	0.0000	neu
1339	Tom Suozzi	False	4	0.2942	pos
1607	Rep. Ruben J. Kihuen	False	3	0.0000	neu
1681	Rep. Ruben J. Kihuen	False	4	0.6458	pos
1743	Rep. Ruben J. Kihuen	False	4	0.6360	pos
1795	Rep. Ruben J. Kihuen	True	4	0.0000	neu
1811	Rep. Donald McEachin	False	3	0.6114	pos
1938	Rep. Donald McEachin	False	3	0.6124	pos
1957	Rep. Donald McEachin	False	2	0.0000	neu
2202	Rep. Charlie Crist	False	4	0.5707	pos
2275	Rep. Charlie Crist	False	3	0.6114	pos
...	...	...	...	...	...
83642	NaN	False	4	0.6114	pos

```
83825                NaN      False          4           0.4199       pos
84060                NaN      False          2           0.5093       pos
84122                NaN       True          4           0.0000       neu
84188                NaN      False          2           0.0000       neu
84331                NaN      False          4           0.6114       pos
84354                NaN      False          3           0.6114       pos
84429                NaN      False          3           0.6114       pos
84445                NaN      False          1           0.0000       neu
84526                NaN      False          3           0.6360       pos
84681                NaN      False          3           0.5707       pos
84736                NaN      False          3           0.6114       pos
84772                NaN       True          2           0.0000       neu
84787                NaN      False          4           0.0000       neu
84812                NaN       True          2           0.0000       neu
85051                NaN      False          4           0.4404       pos
85063                NaN      False          4           0.0000       neu
85115                NaN      False          4           0.6588       pos
85149                NaN      False          4           0.0000       neu
85224                NaN      False          3           0.5574       pos
85236                NaN      False          3           0.0000       neu
85509                NaN      False          3           0.5719       pos
85575                NaN      False          3           0.0000       neu
85608                NaN      False          4           0.0000       neu
85612                NaN      False          3           0.3182       pos
85707                NaN      False          3           0.0000       neu
85836                NaN      False          3           0.0000       neu
85902                NaN       True          2           0.0000       neu
85904                NaN      False          2           0.0000       neu
86069                NaN      False          4           0.0000       neu

[1449 rows x 8 columns]
```

In [25]:
```
# Time to tokenize the data to facilitate text analsis!!!
# This is where the fun begins!

from nltk.tokenize import TweetTokenizer
from nltk.corpus import stopwords
tt = TweetTokenizer()

demTweets = tweetsDf[tweetsDf['Party'] == 'Democrat']['Tweet'].apply(tt.tokenize)
repTweets = tweetsDf[tweetsDf['Party'] == 'Republican']['Tweet'].apply(tt.tokenize)

tokenizedTweets = demTweets.append(repTweets)
```

In [26]:
```
# Confirm tokens are formed correctly.

tokenizedTweets.head()
```

Out[26]:
```
0    [Today, ,, Senate, Dems, vote, to, #SaveTheInt...
```

```
1    [@WinterHavenSun, :, Winter, Haven, resident, ...
2    [@NBCLatino, :, ., @RepDarrenSoto, noted, that...
3    [@NALCABPolicy, :, Meeting, with, @RepDarrenSo...
4    [@Vegalteno, :, Hurricane, season, starts, on,...
Name: Tweet, dtype: object
```

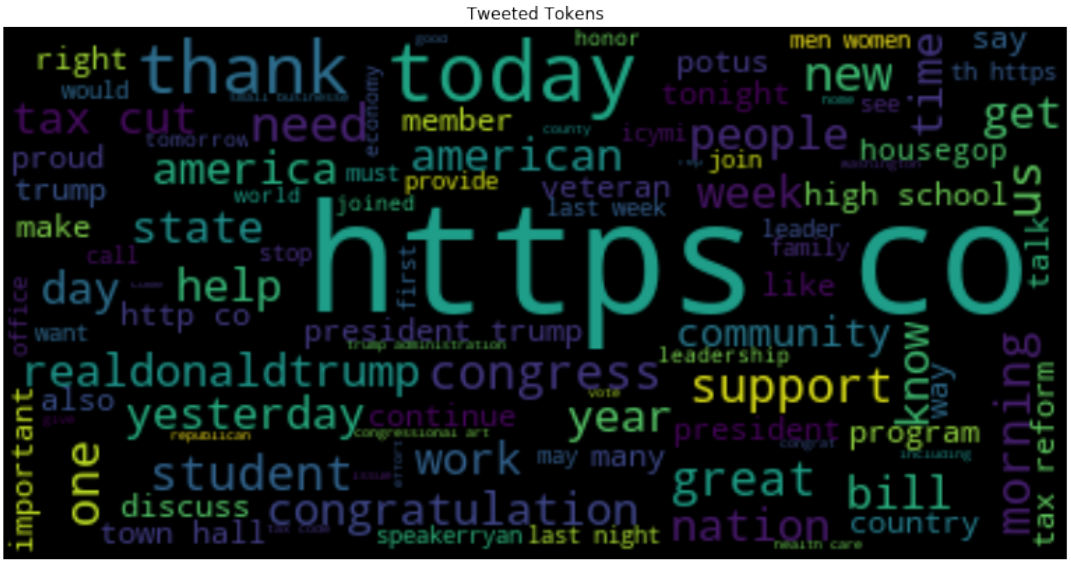
In [27]:
```
#  Create a wordcloud to help viz the text data
# Great step for textual EDA
# Don;t forget to remove stop words!!

import wordcloud

from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

stopWords = stopwords.words('english')
stopWords.append('https')
stopWords.append('Act')
stopWords.append('bill')
stopWords.append('country')
stopWords.append('co')
stopWords.append('today')
stopWords.append('thank')
stopWords.append('american')
stopWords.append('great')
```

In [28]:
```
# Apply stemmer and or lemmatizer!!

from nltk.stem import SnowballStemmer
stemmer = SnowballStemmer("english")

mergedTweets = []
stemmedTokenizedTweets = []
for tweet in tokenizedTweets:
    newT = []
    for t in tweet:
        t = t.strip()
        mergedTweets.append(t.lower())
        newT.append(stemmer.stem(t.lower()))
    stemmedTokenizedTweets.append(" ".join(newT))
#mergedTweets
```

In [29]:
```
allTweets = " ".join(mt for mt in mergedTweets)
allStemmedTweets = " ".join(stt for stt in stemmedTokenizedTweets)
#allTweets
```

In [30]:
```
# After data cleaning, create the word cloud!!
# First we will show the word cloud prior to the stopword removal
# Note there are many useless words, eg, https. thats why we want to remove :)
```

```
wordcloud = WordCloud(background_color='black', max_words=100, stopwords=stopwords.wo
plt.rcParams['figure.figsize'] = [14, 7]
plt.imshow(wordcloud, interpolation='bilinear')
plt.title("Tweeted Tokens")
plt.axis("off")
plt.show()
```



In [31]: *# Show wordcloud after stop word removal*

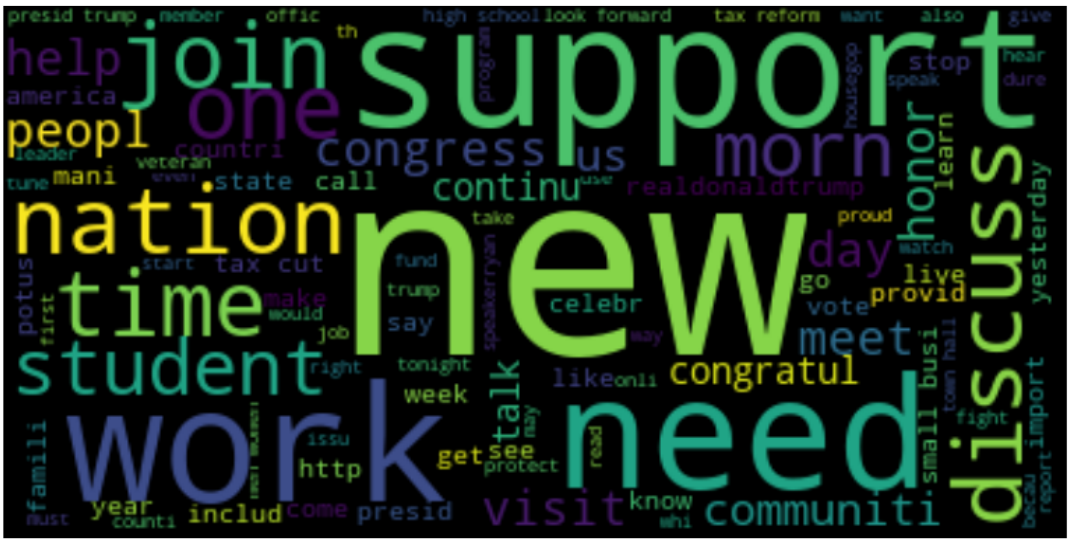
```
wordcloud = WordCloud(background_color='black', max_words=100, stopwords=stopWords).g
plt.rcParams['figure.figsize'] = [14, 7]
plt.imshow(wordcloud, interpolation='bilinear')
plt.title("Tweeted Tokens")
plt.axis("off")
plt.show()
```

Tweeted Tokens

In [32]: *# show wordcloud after stemmer!*

```
wordcloud = WordCloud(background_color='black', max_words=100, stopwords=stopWords).g
plt.rcParams['figure.figsize'] = [14, 7]
plt.imshow(wordcloud, interpolation='bilinear')
plt.title("Stemmed Tweeted Tokens")
plt.axis("off")
plt.show()
```

Stemmed Tweeted Tokens

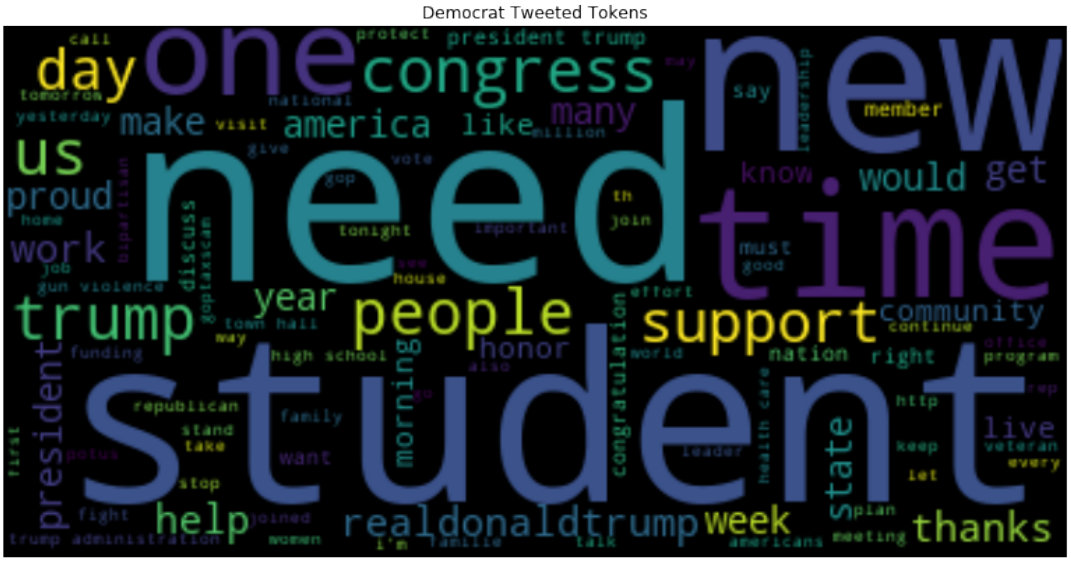


In [33]:
```
# Word clouds helps to viz the text frequency, but we can collect the actual frequenc

mergedTweetsNoStops = []
mergedTweetsStemmed = []
for w in mergedTweets:
    if stopWords.count(w.lower()) == 0:
        mergedTweetsNoStops.append(w)
        mergedTweetsStemmed.append(stemmer.stem(w))

#for key,val in freq.items():
#    print (str(key) + ':' + str(val))
```

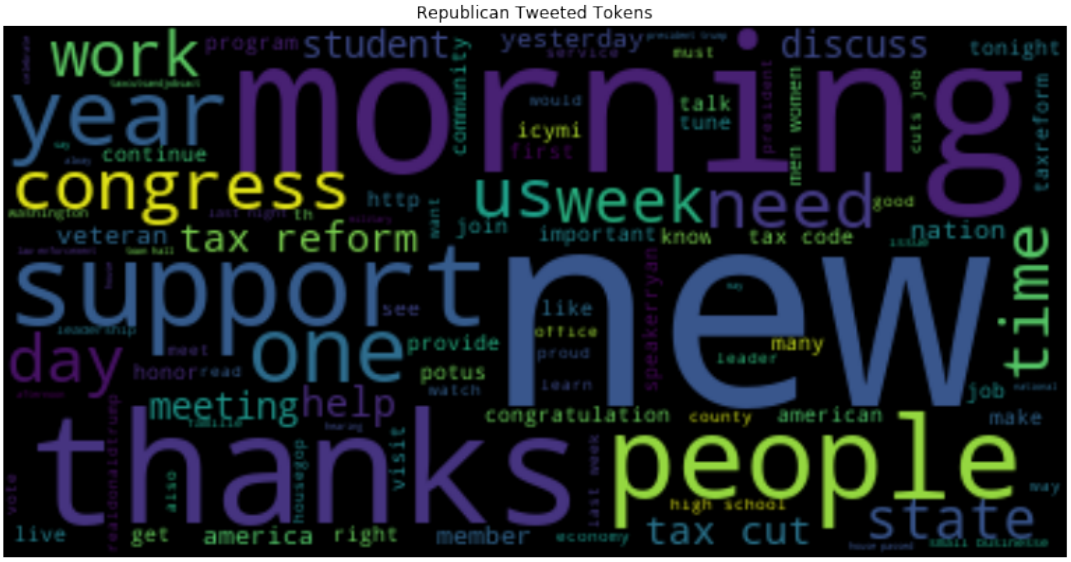
In [34]:
```
# Collect frequent words

freq = nltk.FreqDist(mergedTweetsNoStops)
print(freq.most_common(25))
print()

freq2 = nltk.FreqDist(mergedTweetsStemmed)
print(freq2.most_common(25))
```

```
[('', 59426), ('.', 59241), (',', 46120), (':', 30620), ('', 14859), ('!', 11874), ('&', 9639)

[('', 59426), ('.', 59241), (',', 46120), (':', 30620), ('', 14859), ('!', 11874), ('&', 9639)
```

In [35]:
```
# Create word cloud for Dems (so we can compare to Reps)

mergedDemTweets = []
for tweet in demTweets:
    for t in tweet:
        if stopWords.count(t.lower()) == 0:
            mergedDemTweets.append(t.lower())

allDemTweets = " ".join(mt for mt in mergedDemTweets)

wordcloud = WordCloud(background_color='black', max_words=100, stopwords=stopWords).g
plt.rcParams['figure.figsize'] = [14, 7]
plt.imshow(wordcloud, interpolation='bilinear')
plt.title("Democrat Tweeted Tokens")
plt.axis("off")
plt.show()
```



In [36]:
```
# Create word cloud for Reps

mergedRepTweets = []
for tweet in repTweets:
    for t in tweet:
        if stopWords.count(t.lower()) == 0:
            mergedRepTweets.append(t.lower())

allRepTweets = " ".join(mt for mt in mergedRepTweets)

wordcloud = WordCloud(background_color='black', max_words=100, stopwords=stopWords).g
plt.rcParams['figure.figsize'] = [14, 7]
plt.imshow(wordcloud, interpolation='bilinear')
plt.title("Republican Tweeted Tokens")
plt.axis("off")
plt.show()
```




```
In [37]: tokens = [t for t in tokenizedTweets]
         len(tokens)

Out[37]: 86460

In [38]: # OK -- Looks like our preprocessing steps were successful. Time to embed (vectorize)
         # Use count vectorizer to do this

         from sklearn.feature_extraction.text import CountVectorizer
         from sklearn.feature_extraction.text import TfidfVectorizer

In [39]: # using count vectorizer to embed. Here we use unigrams and bigrams.
         # Be sure to note stopwords and check other relevant parameters ... see docs

         #  unigram boolean vectorizer, set minimum document frequency to 5
         unigram_bool_vectorizer = CountVectorizer(encoding='latin-1', binary=True, min_df=5, s

         #  unigram term frequency vectorizer, set minimum document frequency to 5
         unigram_count_vectorizer = CountVectorizer(encoding='latin-1', binary=False, min_df=5

         #  bigram term frequency vectorizer, set minimum document frequency to 5
         bigram_count_vectorizer = CountVectorizer(encoding='latin-1', ngram_range=(2,3), min_c

         #  unigram tfidf vectorizer, set minimum document frequency to 5
         unigram_tfidf_vectorizer = TfidfVectorizer(encoding='latin-1', use_idf=True, min_df=5

In [40]: # Perform embedding using fit_transform
         # Raw tweets first, then preprocessed results for comparison purposes
```

```
        unigram_bool_v = unigram_bool_vectorizer.fit_transform(tweetsDf['Tweet'].values)
        unigram_count_v = unigram_count_vectorizer.fit_transform(tweetsDf['Tweet'].values)
        bigram_count_v = bigram_count_vectorizer.fit_transform(tweetsDf['Tweet'].values)
        unigram_tfidf_v = unigram_tfidf_vectorizer.fit_transform(tweetsDf['Tweet'].values)

        print(unigram_bool_v.shape)
        print(unigram_count_v.shape)
        print(bigram_count_v.shape)
        print(unigram_tfidf_v.shape)

        unigram_bool_df = pd.DataFrame(columns=unigram_bool_vectorizer.get_feature_names(), da
        unigram_count_df = pd.DataFrame(columns=unigram_count_vectorizer.get_feature_names(),
        bigram_count_df = pd.DataFrame(columns=bigram_count_vectorizer.get_feature_names(), da
        unigram_tfidf_df = pd.DataFrame(columns=unigram_tfidf_vectorizer.get_feature_names(),
```

```
(86460, 200)
(86460, 200)
(86460, 200)
(86460, 200)
```

In [41]:
```
# Viz results of embedding. Note this will (likely) be a very sparse matrix.

unigram_tfidf_df.head()
```

Out[41]:

	000	10	2018	across	act	action	administration	also	always	america	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	...	week	women	work	workers	working	world	would	year	years	\
0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	yesterday
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0

[5 rows x 200 columns]

In [42]:
```
# Now lets embed the preprocessed tweets

unigram_bool_vs = unigram_bool_vectorizer.fit_transform(stemmedTokenizedTweets)
unigram_count_vs = unigram_count_vectorizer.fit_transform(stemmedTokenizedTweets)
bigram_count_vs = bigram_count_vectorizer.fit_transform(stemmedTokenizedTweets)
unigram_tfidf_vs = unigram_tfidf_vectorizer.fit_transform(stemmedTokenizedTweets)

print(unigram_bool_vs.shape)
print(unigram_count_vs.shape)
print(bigram_count_vs.shape)
print(unigram_tfidf_vs.shape)

unigram_bool_dfs = pd.DataFrame(columns=unigram_bool_vectorizer.get_feature_names(), c
unigram_count_dfs = pd.DataFrame(columns=unigram_count_vectorizer.get_feature_names()
bigram_count_dfs = pd.DataFrame(columns=bigram_count_vectorizer.get_feature_names(), c
unigram_tfidf_dfs = pd.DataFrame(columns=unigram_tfidf_vectorizer.get_feature_names()
```

```
(86460, 200)
(86460, 200)
(86460, 200)
(86460, 200)
```

In [43]:
```
unigram_tfidf_dfs.head()
```

Out[43]:

	000	2018	across	act	action	address	administr	alway	america	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	announc	...	way	week	whi	wish	women	work	world	would	year	\
0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	yesterday
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0

[5 rows x 200 columns]

In [44]:
```
# Organize the results into a dataframe
```

```
unigram_bool_df2 = tweetsDf.join(unigram_bool_df)
unigram_count_df2 = tweetsDf.join(unigram_count_df)
bigram_count_df2 = tweetsDf.join(bigram_count_df)
unigram_tfidf_df2 = tweetsDf.join(unigram_tfidf_df)
```

In [45]:
```
bigram_count_df2.head()
```

Out[45]:

	Party	Handle	Tweet	Name	IsRetweet	WordCount	sentiment_score	sentiment	2018 congressional	2018 congressional art	...	water crisis	white house	women uniform	work together	working families	working hard	year old	years ago	years later	young people
0	Democrat	RepDarrenSoto	Today, Senate Dems vote to #SaveTheInternet. P...	US Rep. Darren Soto	False	17	0.7003	pos	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Democrat	RepDarrenSoto	@WinterHavenSun: Winter Haven resident / Alta ...	US Rep. Darren Soto	True	18	0.0000	neu	0	0	...	0	0	0	0	0	0	0	0	0	0
2	Democrat	RepDarrenSoto	@NBCLatino: .@RepDarrenSoto noted that Hurrica...	US Rep. Darren Soto	True	18	-0.4404	neg	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Democrat	RepDarrenSoto	@NALCABPolicy: Meeting with @RepDarrenSoto . T...	US Rep. Darren Soto	True	17	0.4404	pos	0	0	...	0	0	0	0	0	0	0	0	0	0
4	Democrat	RepDarrenSoto	@Vegalteno: Hurricane season starts on June 1s...	US Rep. Darren Soto	True	13	0.0000	neu	0	0	...	0	0	0	0	0	0	0	0	0	0

[5 rows x 208 columns]

In [46]:
```
unigram_bool_dfs2 = tweetsDf.join(unigram_bool_dfs)
unigram_count_dfs2 = tweetsDf.join(unigram_count_dfs)
bigram_count_dfs2 = tweetsDf.join(bigram_count_dfs)
unigram_tfidf_dfs2 = tweetsDf.join(unigram_tfidf_dfs)
```

In [47]:
```
bigram_count_dfs2.head()
```

Out[47]:

	Party	Handle	Tweet	Name	IsRetweet	WordCount	sentiment_score	sentiment	00 pm	2018 congression	...	white hous	wish everyon	women uniform	work famili	work hard	work togeth	year ago	year later	year old	young peopl
0	Democrat	RepDarrenSoto	Today, Senate Dems vote to #SaveTheInternet. P...	US Rep. Darren Soto	False	17	0.7003	pos	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Democrat	RepDarrenSoto	@WinterHavenSun: Winter Haven resident / Alta ...	US Rep. Darren Soto	True	18	0.0000	neu	0	0	...	0	0	0	0	0	0	0	0	0	0
2	Democrat	RepDarrenSoto	@NBCLatino: .@RepDarrenSoto noted that Hurrica...	US Rep. Darren Soto	True	18	-0.4404	neg	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Democrat	RepDarrenSoto	@NALCABPolicy: Meeting with @RepDarrenSoto . T...	US Rep. Darren Soto	True	17	0.4404	pos	0	0	...	0	0	0	0	0	0	0	0	0	0
4	Democrat	RepDarrenSoto	@Vegalteno: Hurricane season starts on June 1s...	US Rep. Darren Soto	True	13	0.0000	neu	0	0	...	0	0	0	0	0	0	0	0	0	0

[5 rows x 208 columns]