Does the Grey Lady treat everyone equally?

Quinn Underriner

Introduction:

This era is marked by heightened scrutiny of the authority of news sources. As distribution methods of information have democratized through the proliferation of social media, there is a heightened cultural debate about what constitutes objectivity and truth. This inquiry is not just for the new media newcomers but also for venerable institutions such as the New York Times ("Fake News", as called by one popular public personality). In response, the New York Times, the paper of record, the liberal news source with the largest readership and highest number of Pulitzer Prizes,[1] has positioned itself as a beacon of truth and the voice for Liberalism in this time of uncertainty.

This paper examines over two thousand articles articles scraped from the New York Times website between April and October 2019 related to the 2020 Democratic primary and conducts two primary forms of analysis:

i. Descriptive natural language processing highlighting how and with what frequency the New York Times has covered democratic nominees in the primary

ii. Study if there are any major discrepancy between (as measured through polling and ANES data) the popularity of candidates in the general population of likely Democratic Party voters and how favorably they are discussed in the New York Times, and if a discrepancy exists, to see if this bias happens on ideological lines. While the New York Times has no obligation to mirror the population, it is useful to see how it deviates from the average "liberal", here loosely defined as the average Democratic Party voter.

---

[1] "Pulitzer Prizes," The New York Times Company. Retrieved December 3, 2019.

The bias element of the second claim is one that has been leveled at the media writ large by the Bernie Sanders campaign. Nina Turner, the Co-National Chair for the Presidential Campaign of Bernie Sanders, recently was interviewed saying "the Bernie blackout is real, its not a figment of our imagination… even though he is polling very high, either number or number 2 [in the polls]… these networks are wired to only care about the upper middle class and the ultra wealthy".[2] She goes on to note that he has raised more money from more individuals than any other candidate[3] and reached one million individual donors before any other candidate this September. To examine this claim this paper will create a measure of ideology among the Democratic nominee hopefuls.

Sources of Data:

I used the Selenium and Beautiful Soup packages to scrape the New York Times website for the phrase "2020 election" across all of their articles (from all categories: politics, op-ed, business, etc.) between May and October of 2019. I gathered 2,100 articles.

search_term	title	date	url	text
2020 Election	Do More Candidates Frighten You?	Oct. 31	https://www.nytimes.com/2019/10/31/us/politics...	On Politics We asked, you answered: Is there a...

This resulted in a data-frame that looked like the above, with the article headline, the date of publication, the url, and the full text as features. There is some risk that coverage of the election will be missed if an article does not contain the phrase "2020 Election", but with 2,100

[2] "Nina Turner talks Sanders media coverage", The Hill. Retrieved December 10, 2019, https://thehill.com/hilltv/rising/473191-nina-turner-talks-sanders-media-coverage

[3] "2020 Presidential Race", Center for Responsive Politics, Retrieved December 9, 2019 https://www.opensecrets.org/2020-presidential-race

articles there should be enough material to capture editorial preferences. Further, while this process might pick up articles that are only tangentially related to the election, as discussed below, the final text used in analysis is further filtered for mentions of specific candidates, so ultimately its likely irrelevant article will not be used. While of course the New York Times will employee writers across the ideological spectrum, I believe the size of this sample will be able to capture if writers of a certain bent are more frequently given assignments.

The articles' text was preprocessed to make all the text lowercase, and stripped of punctuation and stopwords (these are words like "the" that do not add to our understating of the underlying corpus). I then used what is the most common stemming procedure, Porter's algorithm, to produce a list of stemmed words. This algorithm works to remove suffixes to standardize related words and return a comparable root[4] (e.g., "house", "houses" and "housing" would all become "hous"; they are all referring to the same object and should be treated as such). After all of this preprocessing has occurred there are roughly one and a half million words to be analyzed.

To map words to candidates I took the candidates last name (being careful to search for this name in its stemmed form) and then stored the ten words before and after this keyword in a nested list.

Across the corpus there were the following number of mentions per candidate:

Biden: 4797 mentions

Warren: 2287 mentions

Bernie: 1465 mentions

---

[4] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. 2008. Retrieved December 5th, 2019, https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Harris: 1026

Buttigieg: 660 mentions

Klobuchar: 533

Beto: 368 mentions

These mentions were further broken down by month so that they could be mapped to polling data.

For polling data, I used data from Real Clear Politics (RCP),[5] a polling aggregator, to find monthly average polling for each candidate across a variety of polls. RCP relies more heavily on some sources that others, with the largest number of polls coming from Politico/Morning Consult and Economist/YouGov, as seen in the chart below.

Politico/Morning Consult	33
Economist/YouGov	26
The Hill/HarrisX	16
Quinnipiac	12
Emerson	8
Harvard-Harris	7
CNN	7
FOX News	7
Monmouth	5
SurveyUSA	4
IBD/TIPP	4
USA Today/Suffolk	3
NBC News/Wall St. Jrnl	3
Reuters/Ipsos	3
LA Times/USC	3
ABC News/Wash Post	3
RCP Average	1

The polls are all taken over a period of roughly 3-5 days. I took the starting month to assign each poll to a given month. This means that there are some polls that straddled months that were assigned to month that they started in.

RCP stops collecting polling for candidates when they drop out of the race, naturally. This meant that while I was able to grab data for Kamala Harris before she left the running, Beto

[5] "Latest 2020 Democratic Presidential Primary Polls", Real Clear Politics, Accessed December 5th, 2019 from https://www.realclearpolitics.com/epolls/latest_polls/democratic_nomination_polls/

O'Rourke had already dropped out of the data set before I scrapped it using a command line tool.[6]

Below is the monthly ranking from each poll sorted by the October rank. Biden is clearly leading across all six months, with Sanders and Warren swapping second and third in September. Klobuchar is notably in perpetual last place.

Candidate	May_Poll_Rank	June_Poll_Rank	July_Poll_Rank	August_Poll_Rank	Sep_Poll_Rank	Oct_Poll_Rank
Biden	1.0	1.0	1.0	1.0	1.0	1.0
Warren	3.0	3.0	3.0	3.0	2.0	2.0
Sanders	2.0	2.0	2.0	2.0	3.0	3.0
Buttigieg	5.0	5.0	5.0	5.0	5.0	4.0
Harris	4.0	4.0	4.0	4.0	4.0	5.0
Klobuchar	6.0	6.0	6.0	6.0	6.0	6.0

The final source of data was the 2018 Pilot Study from the American National Election Studies (ANES).[7] I used the pilot study as the comprehensive study for 2018 has not yet been released (this was the latest data available). The main difference between the pilot study and the full study is the sample size, N=2,500, is about half as large. This survey gathers detailed demographic and political opinion data from a statistically representative portion of Americans. All participants are US citizens over the age of 18.

While there is a wealth of information in the ANES, I used two features in this analysis. First, "ideo5", which is a 5 point (1 being the most liberal, 5 being the most conservative, with a 6th option for "unsure") scale of voters' self-identified political preferences. Second, "vote20cand" which asks "In the 2020 Democratic primary for president, who will you vote for?

---

6 Anthony Bloomer, "Real Clear Politics,"https://pypi.org/project/realclearpolitics/

7 "ANES 2018 Pilot Study Questionnaire Specifications", American National Election Studies, Retrieved December 1, 2019, https://electionstudies.org/wp-content/uploads/2018/12/anes_pilot_2018_questionnaire.pdf

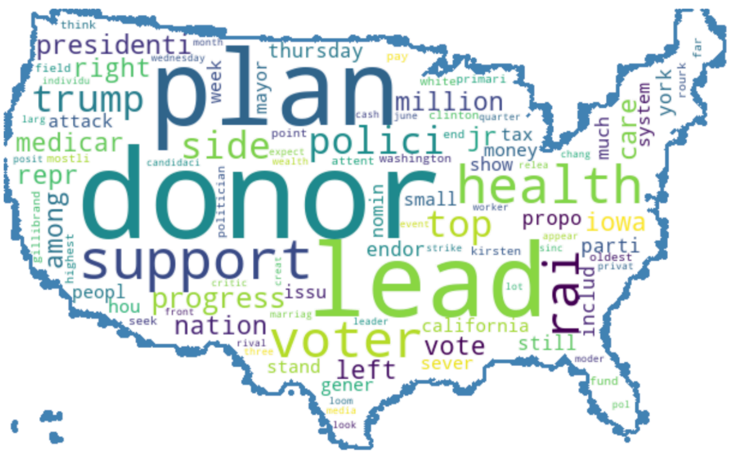
Your best guess is fine” and gives 11 options. Of the 2500 people in the survey, 1243 of the 2500 answered this question (presumably some people didn’t know and a large number are Republicans). Below is the sum of results from “vote20cand”. Compared the poll results, which reflect a later point in time, we see similarity at both the top (with Biden at a comfortable lead and Bernie in second) and at the bottom (Klobuchar), but otherwise there is some shuffling (notably the ascendency of Warren in the polls).

index	vote20cand
Joe Biden	391
Bernie Sanders	246
Beto O'Rourke	183
Kamala Harris	115
Elizabeth Warren	105
Cory Booker	54
Amy Klobuchar	49

It’s important to note that some later entrants to the presidential race, like Pete Buttigieg, are absent from this list (he announced his candidacy April 14, 2019) and therefore is not included in the analysis that involves ANES data. This data is used together to gather a representative sample of popularity of each given candidate in the general population, as well as place candidates on an ideological spectrum. To do this latter analysis it is assumed that who voters say they are voting for is the candidate which most aligns with their values. This seems like a fair assumption, but voters could be factoring in things like “electability” over their own policy preferences. They could also be poorly informed, either about a candidates actual policies or where their own politics land on the US political spectrum. This could slightly skew the results. A final caveat is that this survey was administered between December 5 and December 21 of 2018. New information could have come to light about candidates after this time period that would have changed peoples opinions by the time roughly six months later that the New York Times sentiment analysis started.

Analysis:

Below are two word clouds, made from the entire six month corpus.



Bernie Sanders

Joe Biden

Word clouds can be useful for some initial exploratory analysis of our data. We can see that the scandal involving Hunter Biden on the board of a Ukrainian gas company has dominated Joe Biden's coverage in the New York Times (which is reiterated later in this paper when topic modeling is applied to his corpus), while coverage of Bernie Sanders has a more of a policy focus, especially relating to healthcare, as well as a focus on his number of donors. Simple analysis at this level does not bear out significant bias.

Below are the aggregate sentiment scores for each candidate in their New York Times coverage. These are calculated by creating a list of all the words associated with each person's name and adding up the sentiment score for each word. I used the Affin method to calculate sentiment which takes each word fed to it and returns a numerical score from -5 to 5 based on its perceived positivity or negativity. This is an admittedly naive approach as word can have different meanings in the aggregate than they do alone, but I'm hoping the large size of the sample will smooth out any minor misclassifications.

	sentiment_scores	sentiment_rank
Warren	2262.0	1.0
Biden	1491.0	2.0
Sanders	1382.0	3.0
klobuchar	678.0	4.0
beto	448.0	5.0
harris	417.0	6.0
Buttigieg	330.0	7.0

I used Afinn over Bing, another popular method, because I wanted more granularity in the scores so I would be more likely to register differences between the candidates. Bing just produces a score of -1 (for negative), 0 (for neutral), or 1 (for positive). We can see that by magnitude Warren has had outsize positive coverage. Klobuchar, who is trailing in the polls and the ANES survey, is doing relatively very well. Coverage of Sanders is decently positive.

In the below chart I'm simply calculating the difference between this sentiment rank above and the rank given by the stated voting preference of participants in the ANES survey (rank being created by the number of people intending to vote for that candidate) and I'm then subtracting the sentiment rank from the voting rank. The difference column can be read as the higher the positive value of the difference column number, the relatively lower the coverage received from the NYTimes from the public opinion baseline. Klobuchar and Warren are clear winners, Harris is the largest loser, and Biden and Sanders both have equally slightly less positive than expected coverage.

	Candidate	Sentiment Rank	Voting Rank	Difference
0	Joe Biden	2	1	1.0
1	Bernie Sanders	3	2	1.0
2	Kamala Harris	6	4	2.0
3	Elizabeth Warren	1	5	-4.0
4	Amy Klobuchar	4	7	-3.0

Below is the same analysis, but done at a monthly level using the aggregated Real Clear Politics Polling data. The way to read the "Coverage Gap" column is that if the number is negative the New York Times covers them positively relatively less than their popularity among voters and if the number is positive then the New York Times covers them relatively more glowingly than their popularity among voters.

	diff_oct	diff_sep	diff_may	diff_june	diff_july	diff_aug	Coverage Gap
Biden	-2.0	-2.0	0.0	-1.0	-3.0	-2.0	-10.0
Harris	-1.0	-3.0	0.0	-2.0	0.0	-1.0	-7.0
Buttigieg	-1.0	-0.5	-1.0	0.0	1.0	-2.0	-3.5
Sanders	1.0	1.0	0.0	-2.0	-2.0	0.0	-2.0
Warren	1.0	1.0	0.0	2.0	-1.0	2.0	5.0
Klobuchar	2.0	2.0	1.0	3.0	2.0	2.0	12.0

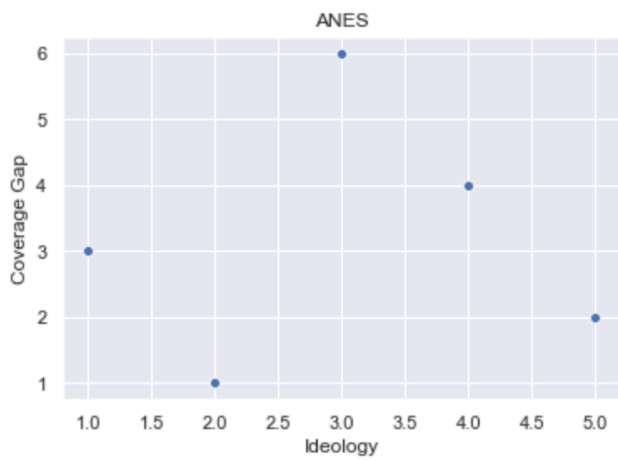
Biden got more overall mentions from the New York Times than the next two largest number of mentions (for Warren and Sanders, respectively) combined. While he's getting a lot of press it's clearly not entirely positive, as shown by his high score; topic model analysis can provide some insight into this. If you set the parameters of the topic model to find just three in the corpus one of the three is related to the corruption allegations involving his son Hunter. Likely much of the negative sentiment comes from related press. It is important to note that as Trump is involved in this scandal as well, this process could also be picking up New York Times' ire directed at Trump and artificially lowering Biden's score.

To gauge the ideology of the candidates, I ran regressions on dummy variables for each of the six self-described points on the ideological spectrum in the ANES. For example, for Bernie Sanders, both the most liberal feature (which had the highest coefficient of any of the candidates) as well as the "I don't know" dummies are the only ones that are statistically significant (see chart below), but for Joe Biden while the most liberal dummy variable is statistically significant, so is the 3rd and 4th point (indicating moderate to conservative leanings) on the spectrum.

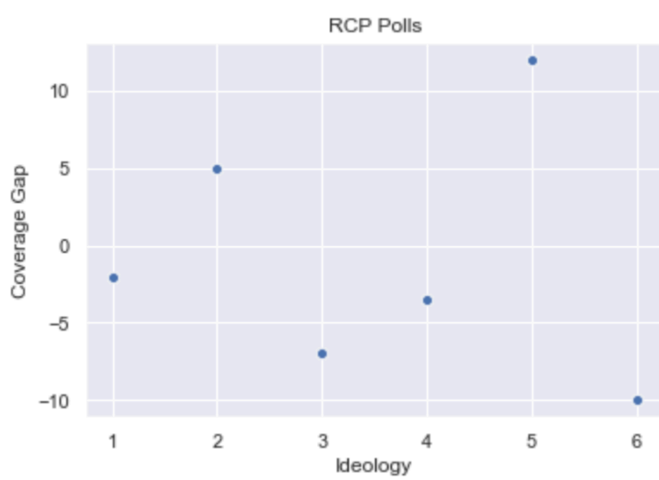
	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1782	0.014	13.122	0.000	0.152	0.205
ideo5_1	0.0579	0.024	2.414	0.016	0.011	0.105
ideo5_2	-0.0166	0.022	-0.745	0.456	-0.060	0.027
ideo5_3	0.0088	0.022	0.404	0.686	-0.034	0.052
ideo5_4	-0.0115	0.038	-0.304	0.761	-0.086	0.063
ideo5_5	-0.0032	0.055	-0.058	0.954	-0.111	0.104
ideo5_6	0.1428	0.040	3.598	0.000	0.065	0.221

I rank ordered the candidates based on the statistical significance of these relative dummy variables, as well as the magnitudes of the related coefficients, with higher coefficients and more significant lower numbered dummies making a candidate more ”liberal”. At the end of this analysis, my ranking of candidates from least liberal to most liberal is: Biden (1), Klobuchar (2), Beto (3), Harris (4), Warren (5) and then Bernie (6).

Below is a scatter plot of these ideology scores vs. the coverage gap figure from above using the ANES rankings.



Doing the same analysis with the RCP polling data:



In both graphs the N is extremely small, but there does seem to be a slight up and to the right trend in both cases (with Biden being a strong outlier, again likely due to the Ukraine scandal), indicating, generally and weakly, that the coverage gap increases as a candidate becomes more liberal.

Ultimately, in response to Nina Turner charge, Senator Sanders does receive significantly less overall coverage than his placement in either the polls or in the ANES rankings. However, the sentiment of the coverage he does receive is close to commensurate with his popularity. The main beneficiary of any alleged New York Times preference for a more centrist candidate is Amy Klobuchar who receives both more total coverage, and especially outsized positive coverage, given the low measures of her popularity. Elizabeth Warren, who is notably the second most leftward candidate according to the above analysis, also has a more pronounced increase in overall coverage and a moderate boost in sentiment. Further and more robust analysis is certainly called for, but these initial results are far from damning or indicative of a strong bias.

In [112]:
```
import pandas as pd
import numpy as np
import nltk
import unicodedata
import sys
import re
import os
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation as LDA

from nltk.stem import PorterStemmer
from afinn import Afinn

import seaborn as sns

from PIL import Image
import matplotlib.pyplot as plt
```

In [288]:
```
# The below data scraped the websites
import time
import requests
import pandas as pd
from bs4 import BeautifulSoup
import json
import string
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
#from webdriver_manager.chrome import ChromeDriverManager


import os

def get_articles(search_term, start_date, end_date):
    pwd = os.getcwd()
    base = "https://www.nytimes.com"
    browser = webdriver.Chrome('/Users/quinnunderriner/Desktop/school/Co
mp American/nytimes_v2_REAL/chromedriver')
    wait = WebDriverWait(browser, 10)
    search_term_url = search_term.replace(' ', '%20')
    search_url = 'https://www.nytimes.com/search?endDate={}&query={}&sor
t=newest&startDate={}'.format(end_date,

search_term_url,

start_date)
    browser.get(search_url)

    while True:
        try:
            time.sleep(1)
            show_more = wait.until(EC.element_to_be_clickable((By.XPATH,
'//button[@type="button"][contains(.,"Show More")]')))
            show_more.click()
        except Exception as e:
                print(e)
                break

    soup = BeautifulSoup(browser.page_source,'lxml')
    search_results = soup.find('ol', {'data-testid':'search-results'})

    articles = []
    links = search_results.find_all('a')
    for link in links:
        link_url = link['href']

        try:
            title = link.find('h4').text
            date = link.find_next('time').text
            print(date + ': '+ title + '\n')

            response = requests.get(base + link_url)
            soup_link = BeautifulSoup(response.text, 'html.parser')
```

```
            scripts = soup_link.find_all('script')
            for script in scripts:
                if 'window.__preloadedData = ' in script.text:
                    jsonStr = script.text
                    jsonStr = jsonStr.split('window.__preloadedData = ')
[-1]
                    jsonStr = jsonStr.rsplit(';',1)[0]

                    jsonData = json.loads(jsonStr)

                    article = []
                    for k, v in jsonData['initialState'].items():
                        w=1
                        try:
                            if v['__typename'] == 'TextInline':
                                article.append(v['text'])
                                #print (v['text'])
                        except:
                            continue
                    article = [ each.strip() for each in article ]
                    article = ''.join([('' if c in string.punctuation el
se ' ')+c for c in article]).strip()
            articles.append([search_term, title, date, base+link_url, ar
ticle])
        except:
            continue

    print("Complete")
    df = pd.DataFrame(articles, columns=['search_term', 'title', 'date',
'url', 'text'])
    df.to_csv('{}_{}_{}.csv'.format(search_term.replace(' ', '_'), start
_date, end_date), index=False)
    browser.quit()
    return df

#Called the scraping function this way, it writes the output of each bat
ch to csvs and I put each piece together below
df = get_articles('2020_Election', '20191016', '20191031')
```

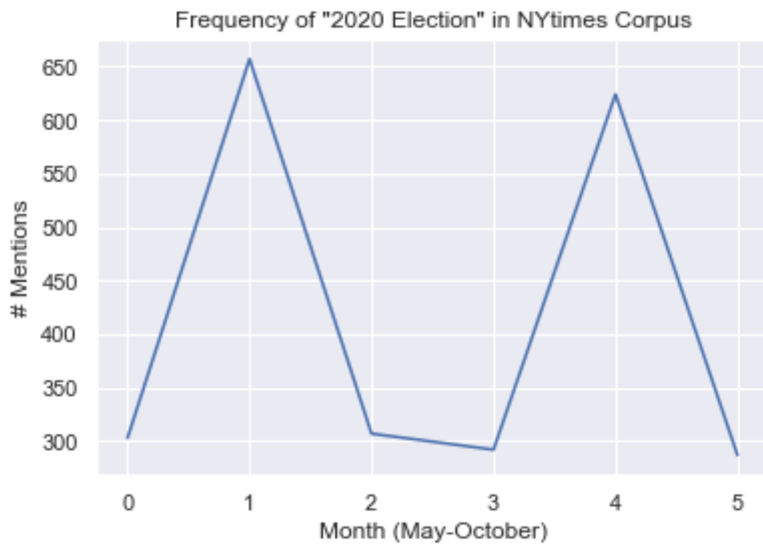
In [ ]:
```
# the below function process text

def keep_chr(char):
    return (unicodedata.category(char).startswith('P'))

PUNCTUATION = " ".join(
    [chr(i) for i in range(sys.maxunicode) if keep_chr(chr(i))])

stop_words = set(stopwords.words('english'))

def preprocess(df):
    df = df.lower()
    df = re.sub(r'\d+', '', df)
    porter = PorterStemmer()

    clean_df = []
    df_split = df.split()
    for i in df_split:
        i = i.strip(PUNCTUATION)
        if i in stop_words:
            continue
        if len(i) == 0:
            continue
        i = porter.stem(i)
        clean_df.append(i)
    return clean_df

def clean_row_level(x):
    return preprocess(x["text"])
```

In [3]:
```
df["cleaned"] = df.apply(lambda x: clean_row_level(x), axis=1)
#after cleaning and writing to csv using the above functions, this loads the saved full corpus
df = pd.read_csv("latest_full_articles.csv")
```

In [ ]:
```
#this is the code that creates wordclouds
def create_cloud_candidates(df):
    # https://www.datacamp.com/community/tutorials/wordcloud-python
    stop_words = set(stopwords.words('english'))
    #add specific stopwords plus names of candidates
    stop_words.update(["would","said","mr","presid","democrat","new","li
ke","elect","year","mrs", \
                       "american","also","could","whether","day","go","us
e","two","one","campaign", \
                       "call","even","say","get","may","make","come","cam
paign","state","polit", \
                       "time","ms","call","elizabeth","warren","joe","bid
en","pete","buttigieg","bernie", \
                       "sander","berni","debat","candid","first","second"
,"percent","night","poll", \
                       "vice","vermont","massachusett","joseph","kamala",
"south","bend","senat","harri","race","former","th"])


    text = " ".join(df)

    usa_mask = np.array(Image.open("US_img.png"))
    wordcloud = WordCloud(max_words=100, stopwords=stop_words, \
                background_color="white", collocations=False,\
                mask=usa_mask, contour_width=2, contour_color='steelblu
e').generate(text)

    plt.figure(figsize=(25,20))

    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.show()
```

In [362]:
```
import numpy as np
y = [303,657,307,292,624,287]
x = np.arange(0,len(y))
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt
ax = sns.lineplot(x=x, y=y)
plt.title('Frequency of "2020 Election" in NYtimes Corpus')
plt.xlabel('Month (May-October)')
plt.ylabel('# Mentions')
```

Out[362]:
```
Text(0, 0.5, '# Mentions')
```



In [4]:
```
#standarize the dates
def strip_date(x):
    return x["date"][0:3]
df["date"] = df.apply(lambda x: strip_date(x), axis=1)
```

In [5]:
```
#break out data by month to be able to do month to month comparisons.
df_may = df[df["date"] == "May"]
df_june = df[df["date"] == "Jun"]
df_jul = df[df["date"] == "July"]
df_aug = df[df["date"] == "Aug"]
df_sep = df[df["date"] == "Sep"]
df_oct = df[df["date"] == "Oct"]
```

In [24]:
```
def gimmie_month_rank(df,month):
    """
    returns a dataframe with the sentimetn scores for each candidate in
 a given month

    """
    df = df.text.str.cat()

    df = preprocess(df)

    sanders_list, warren_list, biden_list, buttigieg_list = extact_relat
ed_words(df)

    sanders_list_text = " ".join(map(str, sanders_list))

    warren_list_text = " ".join(map(str, warren_list))

    biden_list_text = " ".join(map(str, biden_list))

    buttigieg_list_text = " ".join(map(str, buttigieg_list))

    afinn = Afinn()

    biden_score = afinn.score(biden_list_text)
    sanders_score = afinn.score(sanders_list_text)
    buttigieg_score = afinn.score(buttigieg_list_text)
    warren_score = afinn.score(warren_list_text)



    df2 = pd.DataFrame(np.array([[biden_score],
                                  [sanders_score],
                                  [buttigieg_score],
                                  [warren_score]]), columns=[month + "_s
entiment_score"],index=["Biden","Sanders","Warren","Buttigieg"])

    return df2
```

In [33]:
```
#this loop creates a dataframe with the sentiment scores for each month
month_list = ["june","jul","aug","sep","oct"]
start = ny.gimmie_month_rank(df_may,"may")
start = start.rank(axis=0, ascending=False)
for i, j in enumerate([df_june, df_jul, df_aug, df_sep, df_oct]):
        new_month = ny.gimmie_month_rank(j,month_list[i])
        new_month = new_month.rank(axis=0, ascending=False)
        start = pd.concat([start, new_month], axis = 1)
```

In [34]:
```
start
```

Out[34]:

	may_sentiment_score	june_sentiment_score	jul_sentiment_score	aug_sentiment_score
Biden	1.0	2.0	4.0	3.0
Sanders	2.0	4.0	4.0	2.0
Buttigieg	6.0	5.0	4.0	7.0
Warren	3.0	1.0	4.0	1.0
Beto	7.0	7.0	4.0	6.0
Harris	4.0	6.0	4.0	5.0
Klobuchar	5.0	3.0	4.0	4.0

In [18]:
```
def generate_ranked_polls(df):
    """
    This merges RCP polls and creates ranked lists from them
    """

    candidate_list = ["Biden","Sanders","Warren","Buttigieg","Harris","K
lobuchar","Date"]
    def strip_date2(x):
        return x["Date"][0:2]

    #polling data grabbed from real clear politics using https://pypi.or
g/project/realclearpolitics/
    #command line tool
    #polls = pd.read_csv("rcp_polls_PLUS.csv")
    df["Date"] = df.apply(lambda x: strip_date2(x), axis=1)

    df = df[candidate_list]
    print(df.columns)
    df = df.groupby("Date").mean()
    df = df.T

    df = df.drop(["4/"], axis=1)

    df["Oct_Poll_Rank"] = df["10"].rank(ascending=False)
    df["Nov_Poll_Rank"] = df["11"].rank(ascending=False)
    df["May_Poll_Rank"] = df["5/"].rank(ascending=False)
    df["June_Poll_Rank"] = df["6/"].rank(ascending=False)
    df["July_Poll_Rank"] = df["7/"].rank(ascending=False)
    df["August_Poll_Rank"] = df["8/"].rank(ascending=False)
    df["Sep_Poll_Rank"] = df["9/"].rank(ascending=False)

    df = df[['May_Poll_Rank','June_Poll_Rank', 'July_Poll_Rank',
       'August_Poll_Rank', 'Sep_Poll_Rank','Oct_Poll_Rank']]

    return df
```

In [9]:
```
#read in polling data
polls = pd.read_csv("rcp_polls_PLUS.csv")
```

In [19]:
```
#create poll rank df
six_candidate_rank_polls = generate_ranked_polls(polls)
```

```
Index(['Biden', 'Sanders', 'Warren', 'Buttigieg', 'Harris', 'Klobucha
r',
       'Date'],
      dtype='object')
```

In [397]:
```
#Concat polling data and sentiment data
final_nytime_sentiment_diff = pd.concat([six_candidate_rank_polls, start
],axis=1)
final_nytime_sentiment_diff = final_nytime_sentiment_diff.dropna(axis=0)
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:2: FutureWarning: Sorting because non-concatenation axis is
not aligned. A future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=Tru
e'.
```

In [36]:
```
final_nytime_sentiment_diff
```

Out[36]:

	May_Poll_Rank	June_Poll_Rank	July_Poll_Rank	August_Poll_Rank	Sep_Poll_Rank	Oc
Biden	1.0	1.0	1.0	1.0	1.0	
Buttigieg	5.0	5.0	5.0	5.0	5.0	
Harris	4.0	4.0	4.0	4.0	4.0	
Klobuchar	6.0	6.0	6.0	6.0	6.0	
Sanders	2.0	2.0	2.0	2.0	3.0	
Warren	3.0	3.0	3.0	3.0	2.0	

In [290]:
```
#calculate the difference between nytimes sentiment and polling data for
each month
final_nytime_sentiment_diff["diff_oct"] = final_nytime_sentiment_diff["o
ct_sentiment_score"] - final_nytime_sentiment_diff["Oct_Poll_Rank"]
final_nytime_sentiment_diff["diff_sep"] = final_nytime_sentiment_diff["s
ep_sentiment_score"] - final_nytime_sentiment_diff["Sep_Poll_Rank"]
final_nytime_sentiment_diff["diff_may"] = final_nytime_sentiment_diff["m
ay_sentiment_score"] - final_nytime_sentiment_diff["May_Poll_Rank"]
final_nytime_sentiment_diff["diff_june"] = final_nytime_sentiment_diff[
"june_sentiment_score"] - final_nytime_sentiment_diff["June_Poll_Rank"]
final_nytime_sentiment_diff["diff_july"] = final_nytime_sentiment_diff[
"jul_sentiment_score"] - final_nytime_sentiment_diff["July_Poll_Rank"]
final_nytime_sentiment_diff["diff_aug"] = final_nytime_sentiment_diff["a
ug_sentiment_score"] - final_nytime_sentiment_diff["August_Poll_Rank"]
```

In [396]:
```
final_nytime_sentiment_diff
```

Out[396]:

	diff_oct	diff_sep	diff_may	diff_june	diff_july	diff_aug	sum
Biden	2.0	2.0	0.0	1.0	3.0	2.0	10.0
Buttigieg	1.0	0.5	1.0	0.0	-1.0	2.0	3.5
Harris	1.0	3.0	0.0	2.0	0.0	1.0	7.0
Klobuchar	-2.0	-2.0	-1.0	-3.0	-2.0	-2.0	-12.0
Sanders	-1.0	-1.0	0.0	2.0	2.0	0.0	2.0
Warren	-1.0	-1.0	0.0	-2.0	1.0	-2.0	-5.0

In [398]:
```
#CORRECT ONE
#calculate the difference between nytimes sentiment and polling data for
each month
final_nytime_sentiment_diff["diff_oct"] =   final_nytime_sentiment_diff[
"Oct_Poll_Rank"] - final_nytime_sentiment_diff["oct_sentiment_score"]
final_nytime_sentiment_diff["diff_sep"] =  final_nytime_sentiment_diff[
"Sep_Poll_Rank"] - final_nytime_sentiment_diff["sep_sentiment_score"]
final_nytime_sentiment_diff["diff_may"] = final_nytime_sentiment_diff["M
ay_Poll_Rank"] - final_nytime_sentiment_diff["may_sentiment_score"]
final_nytime_sentiment_diff["diff_june"] =  final_nytime_sentiment_diff[
"June_Poll_Rank"] - final_nytime_sentiment_diff["june_sentiment_score"]
final_nytime_sentiment_diff["diff_july"] = final_nytime_sentiment_diff[
"July_Poll_Rank"] - final_nytime_sentiment_diff["jul_sentiment_score"]
final_nytime_sentiment_diff["diff_aug"] = final_nytime_sentiment_diff["A
ugust_Poll_Rank"] - final_nytime_sentiment_diff["aug_sentiment_score"]
```

In [399]:
```
sentiment_diff = final_nytime_sentiment_diff[["diff_oct","diff_may","dif
f_sep","diff_july","diff_june","diff_aug"]]
sentiment_diff = sentiment_diff.abs()
sentiment_diff.sum().sum()/36
#Nytimes is on average deviates by 1.3 from the polling data
```

Out[399]:
```
1.3194444444444444
```

In [401]:
```
#sum these differences between sentiment and polling by candidate
final_nytime_sentiment_diff = final_nytime_sentiment_diff[["diff_oct","d
iff_sep","diff_may","diff_june","diff_july","diff_aug"]]
final_nytime_sentiment_diff['sum'] = final_nytime_sentiment_diff.sum(axi
s=1)
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imp
orts until
```

In [402]:
```
final_nytime_sentiment_diff.sort_values(by=['sum'])
```

Out[402]:

	diff_oct	diff_sep	diff_may	diff_june	diff_july	diff_aug	sum
Biden	-2.0	-2.0	0.0	-1.0	-3.0	-2.0	-10.0
Harris	-1.0	-3.0	0.0	-2.0	0.0	-1.0	-7.0
Buttigieg	-1.0	-0.5	-1.0	0.0	1.0	-2.0	-3.5
Sanders	1.0	1.0	0.0	-2.0	-2.0	0.0	-2.0
Warren	1.0	1.0	0.0	2.0	-1.0	2.0	5.0
Klobuchar	2.0	2.0	1.0	3.0	2.0	2.0	12.0

In [427]:
```
sorted_sentiment = final_nytime_sentiment_diff.sort_values(by=['sum'])
```

In [446]:
```
ideology = np.array([6,3,4,1,2,5])
sorted_sentiment['ideology'] = np.array(ideology)
```

In [447]:
```
sorted_sentiment
```

Out[447]:

	diff_oct	diff_sep	diff_may	diff_june	diff_july	diff_aug	sum	ideology
Biden	-2.0	-2.0	0.0	-1.0	-3.0	-2.0	-10.0	6
Harris	-1.0	-3.0	0.0	-2.0	0.0	-1.0	-7.0	3
Buttigieg	-1.0	-0.5	-1.0	0.0	1.0	-2.0	-3.5	4
Sanders	1.0	1.0	0.0	-2.0	-2.0	0.0	-2.0	1
Warren	1.0	1.0	0.0	2.0	-1.0	2.0	5.0	2
Klobuchar	2.0	2.0	1.0	3.0	2.0	2.0	12.0	5

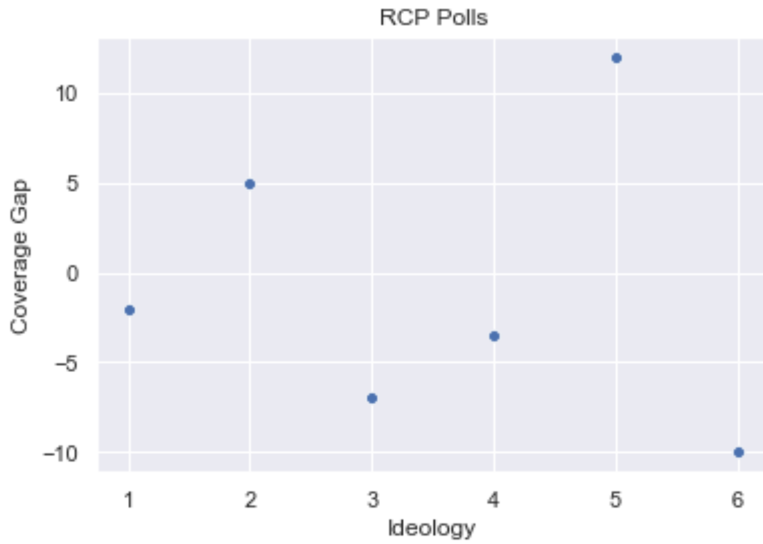
In [456]:
```
sorted_sentiment.rename(columns={"sum": "Coverage Gap"})
```

Out[456]:

	diff_oct	diff_sep	diff_may	diff_june	diff_july	diff_aug	Coverage Gap	ideology
Biden	-2.0	-2.0	0.0	-1.0	-3.0	-2.0	-10.0	6
Harris	-1.0	-3.0	0.0	-2.0	0.0	-1.0	-7.0	3
Buttigieg	-1.0	-0.5	-1.0	0.0	1.0	-2.0	-3.5	4
Sanders	1.0	1.0	0.0	-2.0	-2.0	0.0	-2.0	1
Warren	1.0	1.0	0.0	2.0	-1.0	2.0	5.0	2
Klobuchar	2.0	2.0	1.0	3.0	2.0	2.0	12.0	5

In [459]:
```
ax = sns.scatterplot(x="ideology", y="sum", data=sorted_sentiment)
plt.ylabel('Coverage Gap')
plt.xlabel('Ideology')
plt.title('RCP Polls')
```

Out[459]:
```
Text(0.5, 1.0, 'RCP Polls')
```



In [ ]:

In [88]:
```
#load anes data
anes = pd.read_csv("anes_pilot_2018.csv")
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/IPython/co
re/interactiveshell.py:2785: DtypeWarning: Columns (69,78) have mixed t
ypes. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [89]:
```
# info from here https://electionstudies.org/wp-content/uploads/2018/12/
anes_pilot_2018_questionnaire.pdf
#create a dict so data has candidates names, not numbers
candidate_dict = {1:"Elizabeth Warren",
                 2:"Joe Biden",
                 3:"Kamala Harris",
                 4:"Cory Booker",
                 5:"Bernie Sanders",
                 6:"Kirsten Gillibrand",
                 7:"Deval Patrick",
                 8:"Eric Holder",
                 9:"Chris Murphy",
                 10:"Amy Klobuchar",
                 11:"Beto O'Rourke"}
```

In [90]:
```
#drop unneeded columns from ANES and apply cadidate name dictionary gene
rated above
anes_sub_keep = anes[["vote20cand","pid1r","media1","media2","media3","m
edia4","trustmedia","gender","race","votereg","ideo5","educ","birthyr",
"newsint"]]
anes_sub = anes[["ideo5","vote20cand"]]
anes_sub["vote20cand"] = anes_sub["vote20cand"].map(candidate_dict)
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imp
orts until
```

In [101]:
```
#look at ideological spread
anes_sub['ideo5'].value_counts()
```

Out[101]:
```
 3    685
 4    475
 2    426
 1    326
 6    302
 5    285
-7      1
Name: ideo5, dtype: int64
```

In [91]:
```
#ready data for use in regression by making dummy variables of the ideol
ogy figures and drop the non-responces
spectrum_dummies = pd.get_dummies(anes_sub, columns=['ideo5'])
spectrum_dummies = spectrum_dummies.dropna(subset=['vote20cand'])

spectrum_dummies = pd.get_dummies(spectrum_dummies, columns=['vote20can
d'])
spectrum_dummies = spectrum_dummies[spectrum_dummies["ideo5_-7"] != 1]
```

In [95]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
y = spectrum_dummies["vote20cand_Bernie Sanders"]
X = spectrum_dummies[["ideo5_1","ideo5_2", "ideo5_3", "ideo5_4", "ideo5_
5", "ideo5_6"]]
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()
predictions = model.predict(X)

model.summary()
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/numpy/cor
e/fromnumeric.py:2389: FutureWarning: Method .ptp is deprecated and wil
l be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[95]:

OLS Regression Results

Dep. Variable:	vote20cand_Bernie Sanders	**R-squared:**	0.012
Model:	OLS	**Adj. R-squared:**	0.008
Method:	Least Squares	**F-statistic:**	2.887
Date:	Mon, 09 Dec 2019	**Prob (F-statistic):**	0.0134
Time:	14:12:13	**Log-Likelihood:**	-612.67
No. Observations:	1243	**AIC:**	1237.
Df Residuals:	1237	**BIC:**	1268.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1782	0.014	13.122	0.000	0.152	0.205
ideo5_1	0.0579	0.024	2.414	0.016	0.011	0.105
ideo5_2	-0.0166	0.022	-0.745	0.456	-0.060	0.027
ideo5_3	0.0088	0.022	0.404	0.686	-0.034	0.052
ideo5_4	-0.0115	0.038	-0.304	0.761	-0.086	0.063
ideo5_5	-0.0032	0.055	-0.058	0.954	-0.111	0.104
ideo5_6	0.1428	0.040	3.598	0.000	0.065	0.221

Omnibus:	266.064	**Durbin-Watson:**	2.026
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	466.953
Skew:	1.493	**Prob(JB):**	4.00e-102
Kurtosis:	3.307	**Cond. No.**	2.59e+15

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.31e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

The two statistically signficant predictor of support for Bernie Sanders is self-identification of being the "most liberal" on a 1-6 ideology scale, as well as the most conservative. Given that those listed are already likley democratic voters this is a really interesting finding. The coefficient on most liberal as a predictor is slighly higher for Bernie than Warren, but not by a significant amount

In [387]:
```
y = spectrum_dummies["vote20cand_Elizabeth Warren"]
X = spectrum_dummies[["ideo5_1","ideo5_2", "ideo5_3", "ideo5_4", "ideo5_5", "ideo5_6"]]
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()
predictions = model.predict(X)

model.summary()
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/numpy/cor
e/fromnumeric.py:2389: FutureWarning: Method .ptp is deprecated and wil
l be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[387]:

OLS Regression Results

Dep. Variable:	vote20cand_Elizabeth Warren	**R-squared:**	0.007
Model:	OLS	**Adj. R-squared:**	0.003
Method:	Least Squares	**F-statistic:**	1.640
Date:	Wed, 11 Dec 2019	**Prob (F-statistic):**	0.146
Time:	17:24:44	**Log-Likelihood:**	-168.86
No. Observations:	1243	**AIC:**	349.7
Df Residuals:	1237	**BIC:**	380.5
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0597	0.010	6.282	0.000	0.041	0.078
ideo5_1	0.0549	0.017	3.270	0.001	0.022	0.088
ideo5_2	0.0211	0.016	1.350	0.177	-0.010	0.052
ideo5_3	0.0208	0.015	1.363	0.173	-0.009	0.051
ideo5_4	-0.0041	0.027	-0.156	0.876	-0.056	0.048
ideo5_5	-0.0597	0.038	-1.558	0.120	-0.135	0.015
ideo5_6	0.0267	0.028	0.962	0.336	-0.028	0.081

Omnibus:	743.814	**Durbin-Watson:**	1.938
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4223.309
Skew:	2.957	**Prob(JB):**	0.00
Kurtosis:	9.824	**Cond. No.**	2.59e+15

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.31e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [390]:
```
y = spectrum_dummies["vote20cand_Joe Biden"]
X = spectrum_dummies[["ideo5_1","ideo5_2", "ideo5_3", "ideo5_4", "ideo5_
5", "ideo5_6"]]
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()
predictions = model.predict(X)

model.summary()
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/numpy/cor
e/fromnumeric.py:2389: FutureWarning: Method .ptp is deprecated and wil
l be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[390]:

OLS Regression Results

Dep. Variable:	vote20cand_Joe Biden	**R-squared:**	0.044
Model:	OLS	**Adj. R-squared:**	0.040
Method:	Least Squares	**F-statistic:**	11.37
Date:	Wed, 11 Dec 2019	**Prob (F-statistic):**	9.63e-11
Time:	17:24:57	**Log-Likelihood:**	-782.27
No. Observations:	1243	**AIC:**	1577.
Df Residuals:	1237	**BIC:**	1607.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.2879	0.016	18.497	0.000	0.257	0.318
ideo5_1	-0.1178	0.027	-4.283	0.000	-0.172	-0.064
ideo5_2	-0.0094	0.026	-0.366	0.715	-0.060	0.041
ideo5_3	0.1251	0.025	4.997	0.000	0.076	0.174
ideo5_4	0.1454	0.043	3.348	0.001	0.060	0.231
ideo5_5	0.0621	0.063	0.989	0.323	-0.061	0.185
ideo5_6	0.0825	0.045	1.813	0.070	-0.007	0.172

Omnibus:	1766.083	**Durbin-Watson:**	1.995
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	194.323
Skew:	0.736	**Prob(JB):**	6.36e-43
Kurtosis:	1.740	**Cond. No.**	2.59e+15

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.31e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

For Joe Biden we see a wider spectrum. Buttigieg was not in the ANES data as he did not announce his candincy for president until April 14, 2019.

In [389]:
```
y = spectrum_dummies["vote20cand_Amy Klobuchar"]
X = spectrum_dummies[["ideo5_1","ideo5_2", "ideo5_3", "ideo5_4", "ideo5_5", "ideo5_6"]]
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()
predictions = model.predict(X)

model.summary()
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/numpy/cor
e/fromnumeric.py:2389: FutureWarning: Method .ptp is deprecated and wil
l be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[389]:

OLS Regression Results

Dep. Variable:	vote20cand_Amy Klobuchar	**R-squared:**	0.001
Model:	OLS	**Adj. R-squared:**	-0.003
Method:	Least Squares	**F-statistic:**	0.2497
Date:	Wed, 11 Dec 2019	**Prob (F-statistic):**	0.940
Time:	17:24:55	**Log-Likelihood:**	271.48
No. Observations:	1243	**AIC:**	-531.0
Df Residuals:	1237	**BIC:**	-500.2
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0334	0.007	5.005	0.000	0.020	0.046
ideo5_1	-0.0021	0.012	-0.180	0.857	-0.025	0.021
ideo5_2	0.0112	0.011	1.022	0.307	-0.010	0.033
ideo5_3	0.0056	0.011	0.521	0.602	-0.015	0.027
ideo5_4	0.0111	0.019	0.595	0.552	-0.025	0.048
ideo5_5	-0.0084	0.027	-0.311	0.756	-0.061	0.044
ideo5_6	0.0160	0.019	0.822	0.411	-0.022	0.054

Omnibus:	1174.615	**Durbin-Watson:**	2.037
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	26111.470
Skew:	4.727	**Prob(JB):**	0.00
Kurtosis:	23.367	**Cond. No.**	2.59e+15

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.31e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [388]:
```
y = spectrum_dummies["vote20cand_Kamala Harris"]
X = spectrum_dummies[["ideo5_1","ideo5_2", "ideo5_3", "ideo5_4", "ideo5_
5", "ideo5_6"]]
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()
predictions = model.predict(X)

model.summary()
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/numpy/cor
e/fromnumeric.py:2389: FutureWarning: Method .ptp is deprecated and wil
l be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[388]:

OLS Regression Results

Dep. Variable:	vote20cand_Kamala Harris	**R-squared:**	0.051
Model:	OLS	**Adj. R-squared:**	0.047
Method:	Least Squares	**F-statistic:**	13.37
Date:	Wed, 11 Dec 2019	**Prob (F-statistic):**	1.01e-12
Time:	17:24:54	**Log-Likelihood:**	-191.29
No. Observations:	1243	**AIC:**	394.6
Df Residuals:	1237	**BIC:**	425.3
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0590	0.010	6.096	0.000	0.040	0.078
ideo5_1	0.1355	0.017	7.925	0.000	0.102	0.169
ideo5_2	0.0524	0.016	3.297	0.001	0.021	0.084
ideo5_3	-0.0226	0.016	-1.454	0.146	-0.053	0.008
ideo5_4	-0.0257	0.027	-0.950	0.342	-0.079	0.027
ideo5_5	-0.0340	0.039	-0.871	0.384	-0.111	0.043
ideo5_6	-0.0466	0.028	-1.650	0.099	-0.102	0.009

Omnibus:	643.887	**Durbin-Watson:**	1.961
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2841.036
Skew:	2.587	**Prob(JB):**	0.00
Kurtosis:	8.299	**Cond. No.**	2.59e+15

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.31e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [377]:
```
y = spectrum_dummies["vote20cand_Beto O'Rourke"]
X = spectrum_dummies[["ideo5_1","ideo5_2", "ideo5_3", "ideo5_4", "ideo5_
5", "ideo5_6"]]
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()
predictions = model.predict(X)

model.summary()
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/numpy/cor
e/fromnumeric.py:2389: FutureWarning: Method .ptp is deprecated and wil
l be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[377]:

OLS Regression Results

Dep. Variable:	vote20cand_Beto O'Rourke	**R-squared:**	0.037
Model:	OLS	**Adj. R-squared:**	0.033
Method:	Least Squares	**F-statistic:**	9.569
Date:	Tue, 10 Dec 2019	**Prob (F-statistic):**	5.69e-09
Time:	20:55:25	**Log-Likelihood:**	-450.51
No. Observations:	1243	**AIC:**	913.0
Df Residuals:	1237	**BIC:**	943.8
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0862	0.012	7.235	0.000	0.063	0.110
ideo5_1	0.1013	0.021	4.810	0.000	0.060	0.143
ideo5_2	0.1338	0.020	6.832	0.000	0.095	0.172
ideo5_3	0.0281	0.019	1.464	0.144	-0.010	0.066
ideo5_4	-0.0418	0.033	-1.257	0.209	-0.107	0.023
ideo5_5	-0.0612	0.048	-1.274	0.203	-0.156	0.033
ideo5_6	-0.0739	0.035	-2.121	0.034	-0.142	-0.006

Omnibus:	396.037	**Durbin-Watson:**	1.938
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	875.984
Skew:	1.864	**Prob(JB):**	6.06e-191
Kurtosis:	4.738	**Cond. No.**	2.59e+15

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.31e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

For someone more centrist like Klobuchar, none of the self-identified ideological points

In [359]:
```
#look at total number of people across ANES data that said they would vo
te for a given candidate
anes_rank = pd.DataFrame(anes_sub.vote20cand.value_counts()).reset_index
()
```

In [360]:
```
anes_rank
```

Out[360]:

	index	vote20cand
0	Joe Biden	391
1	Bernie Sanders	246
2	Beto O'Rourke	183
3	Kamala Harris	115
4	Elizabeth Warren	105
5	Cory Booker	54
6	Amy Klobuchar	49
7	Kirsten Gillibrand	38
8	Chris Murphy	26
9	Eric Holder	22
10	Deval Patrick	14

In [115]:
```
nytimes_full_text = df.text.str.cat()
nytimes_full_titles = df.title.str.cat()
nytimes_full_text = preprocess(nytimes_full_text)
nytimes_full_titles = preprocess(nytimes_full_titles)
```

In [146]:
```
def extact_related_words(df):
    """
    grab words surrounding a candidates name to create corpus for each c
andidate
    """
    sanders_list = []
    warren_list = []
    biden_list = []
    buttigieg_list = []
    beto_list = []
    harris_list = []
    klobuchar_list = []


    for index, word in enumerate(df):
        if word == "sander":
            sanders_list.append(df[index-10:index+10])
        if word == "warren":
            warren_list.append(df[index-10:index+10])
        if word == "biden":
            biden_list.append(df[index-10:index+10])
        if word == "buttigieg":
            buttigieg_list.append(df[index-10:index+10])
        if word == "beto":
            beto_list.append(df[index-10:index+10])
        if word == "harri":
            harris_list.append(df[index-10:index+10])
        if word == "klobuchar":
            klobuchar_list.append(df[index-10:index+10])




    return sanders_list, warren_list, biden_list, buttigieg_list, beto_l
ist, harris_list, klobuchar_list
```

In [152]:
```
#create corpus of text for each candidate
sanders_list, warren_list, biden_list, buttigieg_list, beto_list, harris
_list, klobuchar_list = extact_related_words(nytimes_full_text)
```

In [153]:

```
    #generate sentiment score for each candidate across all six months
    sanders_list_text = " ".join(map(str, sanders_list))
    warren_list_text = " ".join(map(str, warren_list))
    biden_list_text = " ".join(map(str, biden_list))
    buttigieg_list_text = " ".join(map(str, buttigieg_list))
    beto_list_text = " ".join(map(str, beto_list))
    harris_list_text = " ".join(map(str, harris_list))
    klobuchar_list_text = " ".join(map(str, klobuchar_list))

    afinn = Afinn()

    biden_score = afinn.score(biden_list_text)
    sanders_score = afinn.score(sanders_list_text)
    buttigieg_score = afinn.score(buttigieg_list_text)
    warren_score = afinn.score(warren_list_text)
    buttigieg_score = afinn.score(buttigieg_list_text)
    beto_score = afinn.score(beto_list_text)
    harris_score = afinn.score(harris_list_text)
    klobuchar_score = afinn.score(klobuchar_list_text)


    full_sentiment = pd.DataFrame(np.array([[biden_score],
                                  [sanders_score],
                                  [buttigieg_score],
                                  [warren_score],
                                            [beto_score],[harris_score],
[klobuchar_score]]),
                                            columns=["sentiment_scores"
],
                                            index=["Biden","Sanders","Bu
ttigieg","Warren","beto","harris","klobuchar"])
```

In [154]:

```
#generate rank for sentiment score accross the whole corpus
full_sentiment["sentiment_rank"] = full_sentiment["sentiment_scores"].ra
nk(ascending=False)
```

In [363]:

```
full_sentiment.sort_values(by='sentiment_rank', ascending=True)
```

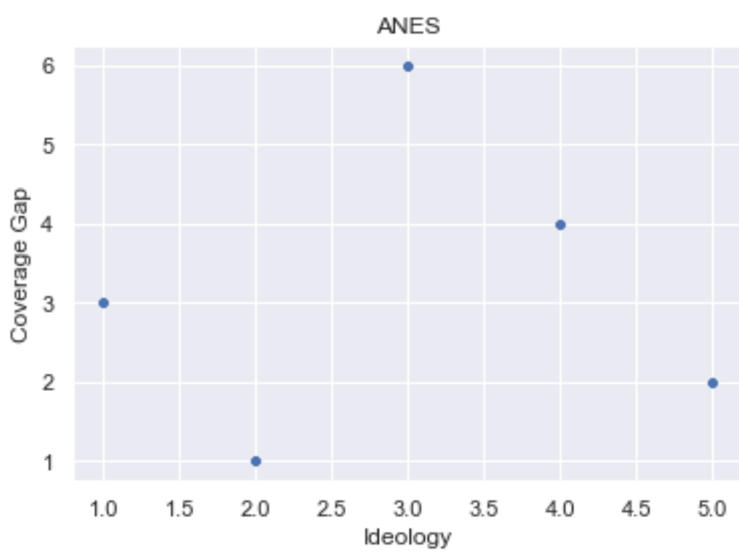
Out[363]:

	index	sentiment_scores	sentiment_rank
3	Elizabeth Warren	2262	1
0	Joe Biden	1491	2
1	Bernie Sanders	1382	3
6	Amy Klobuchar	678	4
4	Beto O'Rourke	448	5
5	Kamala Harris	417	6
2	Pete Buttigieg	330	7

In [372]:
```
#generate DF with candidate level difference between sentiment rank and
 ANES stated voting preference rank
anes_rank = anes_rank[["index","vote20cand"]]
full_comparison = anes_rank.merge(full_sentiment, on="index",how="inner"
)
anes_rank["Voting_rank"] = anes_rank["vote20cand"].rank(ascending=False)
full_comparison = full_comparison[["index","Voting_rank","sentiment_ran
k"]]
full_comparison["difference"] = full_comparison["sentiment_rank"] - full
_comparison["Voting_rank"]
#full_comparison = full_comparison[["index","vote20cand","sentiment_scor
es"]]
full_comparison =  full_comparison[["index","sentiment_rank","Voting_ran
k","difference"]]
full_comparison = full_comparison.rename(columns={"index": "Candidate",
"sentiment_rank":"Sentiment Rank","Voting_rank":"Voting Rank","differenc
e":"Difference"})
full_comparison["Voting Rank"] = full_comparison["Voting Rank"].astype(i
nt)
```

In [458]:
```
ideology2 = np.array([5,1,3,2,4])
full_comparison['ideology'] = np.array(ideology2)
ax = sns.scatterplot(x="ideology", y="sentiment_rank", data=full_compari
son)

plt.xlabel('Ideology')
plt.ylabel('Coverage Gap')
plt.title('ANES')
```

Out[458]: Text(0.5, 1.0, 'ANES')



In [238]:
```
#ready text for use in topic model
test_text = ""
for i in sanders_list:
    for j in i:
        test_text += (j + " ")

test_text = [test_text]
```

In [274]:
```
def text_topic_model(df):
    """
    make text right format for topic models
    """
    new_list = []
    for i in df:
        new_list.append(' '.join(i))
    return new_list
```

In [259]:
```
len(new_list_bernie)
```

Out[259]: 1465

In [247]:
```
from sklearn.feature_extraction import text

stop_words = text.ENGLISH_STOP_WORDS.union("ms","candid","mr","said")
```

In [281]:
```
    stop_words = set(stopwords.words('english'))
    #add specific stopwords plus names of candidates
    stop_words.update(["would","said","mr","presid","booker","democrat",
"new","like","elect","year","mrs", \
                       "american","also","issu","june","moslty","could",
"whether","day","go","use","two","one","campaign", \
                       "call","even","say","get","may","make","come","cam
paign","state","polit", \
                       "time","ms","call","elizabeth","warren","joe","bid
en","pete","buttigieg","bernie", \
                       "sander","berni","rourk","debat","campaign","candi
d","first","second","percent","night","poll", \
                       "vice","vermont","massachusett","joseph","kamala",
"south","bend","jr","senat","harri","race","former","th"])
```

In [273]:

```
#topic modeling for bernie

count_vectorizer = CountVectorizer(stop_words=stop_words)

count_data_bernie = count_vectorizer.fit_transform(new_list_bernie)

def print_topics(model, count_vectorizer, n_top_words):
    words = count_vectorizer.get_feature_names()
    for topic_idx, topic in enumerate(model.components_):
        print("\nTopic #%d:" % topic_idx)
        print(" ".join([words[i]
                        for i in topic.argsort()[:-n_top_words - 1:-1
]]))

number_topics = 7
number_words = 8

# LDA model
lda = LDA(n_components=number_topics, n_jobs=-1)
lda.fit(count_data_bernie)

print("Topics found via LDA:")
print_topics(lda, count_vectorizer, number_words)
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/sklearn/de
composition/online_lda.py:536: DeprecationWarning: The default value fo
r 'learning_method' will be changed from 'online' to 'batch' in the rel
ease 0.20. This warning was introduced in 0.18.
  DeprecationWarning)

Topics found via LDA:

Topic #0:
health care among lead nation medicar gener trump

Topic #1:
voter endors top right look creat team media

Topic #2:
lead show point larg els appear rival onstag

Topic #3:
progress includ sever class much expect york support

Topic #4:
plan presidenti money repres million rais vote tax

Topic #5:
side stand left polici white mostli seek nomin

Topic #6:
donor small individu far mayor moder andrew estim
```

In [276]:
```
#generate list for biden
new_list_biden = text_topic_model(biden_list)
```

In [454]:
```
count_vectorizer = CountVectorizer(stop_words=stop_words)

count_data_biden = count_vectorizer.fit_transform(new_list_biden)

def print_topics(model, count_vectorizer, n_top_words):
    words = count_vectorizer.get_feature_names()
    for topic_idx, topic in enumerate(model.components_):
        print("\nTopic #%d:" % topic_idx)
        print(" ".join([words[i]
                        for i in topic.argsort()[:-n_top_words - 1:-1
]]))

number_topics = 3
number_words = 8

# LDA model
lda = LDA(n_components=number_topics, n_jobs=-1)
lda.fit(count_data_biden)

print("Topics found via LDA:")
print_topics(lda, count_vectorizer, number_words)
```

```
/Users/quinnunderriner/anaconda3/lib/python3.7/site-packages/sklearn/de
composition/online_lda.py:536: DeprecationWarning: The default value fo
r 'learning_method' will be changed from 'online' to 'batch' in the rel
ease 0.20. This warning was introduced in 0.18.
  DeprecationWarning)

Topics found via LDA:

Topic #0:
trump voter support clinton front lead primari iowa

Topic #1:
trump investig son hunter ukrain ukrainian zelenski corrupt

Topic #2:
trump rais fund far donor back attack polici
```