

Media bias

Michael Kuhn

Herman Schaumburg

May 9, 2020

Goals

Primary: Find objective way to score bias in 36 selected news organizations.

Secondary: Estimate the political bias of a given tweet.

Secondary: Heroku deployment.



Top Online News Media Bias Ratings

AllSides media bias ratings are based on multi-partisan, scientific analysis.
All ratings are based on online content only — not TV, print, or radio content.
Ratings do not reflect accuracy or credibility; they speak to perspective only.
Visit AllSides.com to view over 600 media bias ratings.



L LEFT L LEAN LEFT C CENTER R LEAN RIGHT R RIGHT

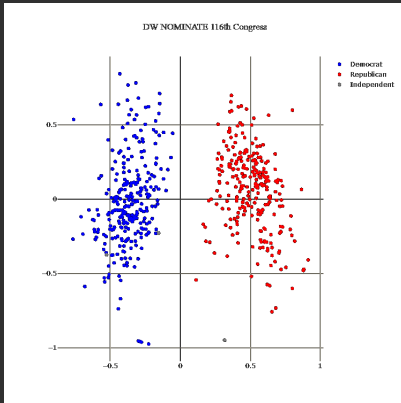
Version 1.1 | AllSides 2019

Objective Media Bias Rankings

www.mediabias.herokuapp.com

Left	Left-Leaning	Moderate	Right-Leaning	Right
      	       	      	       	      

DW-NOMINATE scores



Methodology

1. Gathered tweets from Democratic and Republican law makers (1,266,104) from Alex Litel github repository into SQL database. From media orgs 54,833.

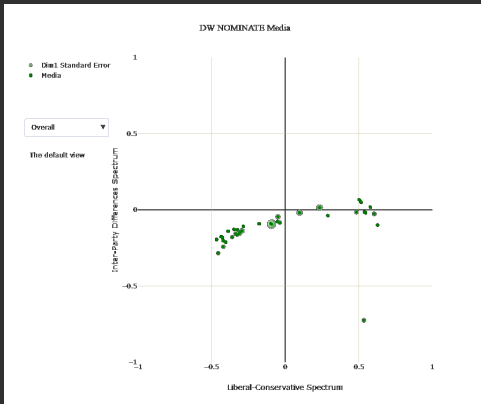
<https://github.com/alexlitel/congresstweets>

2. DW-NOMINATE scores of congress people from voteview

<https://voteview.com/data>

3. Queried tweets of congresspeople for the media domains they tweet from and used the congressperson DW-NOMINATE score to score the media domain.

Media domain scores



Method diagram



$(-1, 0)$

Tweets: A, A, B, B

$$A = \frac{(1,0) + (-1,0) + (-1,0)}{3} = \frac{(-1,0)}{3} = \left(-\frac{1}{3}, 0\right)$$

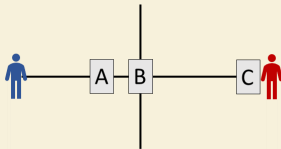


$(1, 0)$

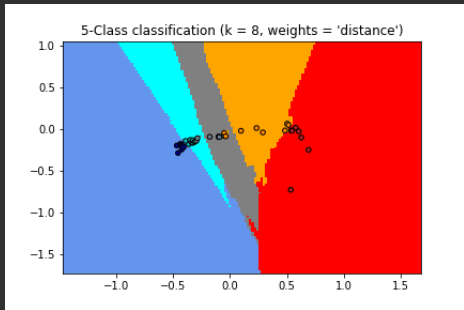
Tweets: A, B, B, C

$$B = \frac{(1,0) + (1,0) + (-1,0) + (-1,0)}{4} = \frac{(0,0)}{4} = (0,0)$$

$$C = \frac{(1,0)}{1} = \frac{(1,0)}{1} = (1,0)$$



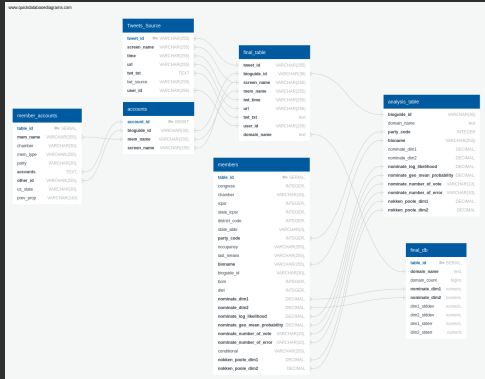
Bias Ranking



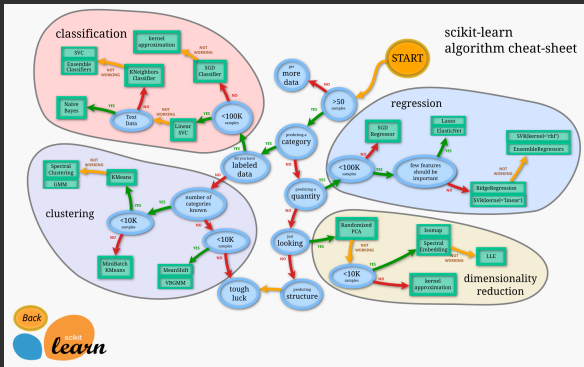
K-Nearest neighbors to classify media domains:

Left, Left-leaning, Moderate, Right-leaning, Right

SQL Schema



ML Model Selection



ML Model

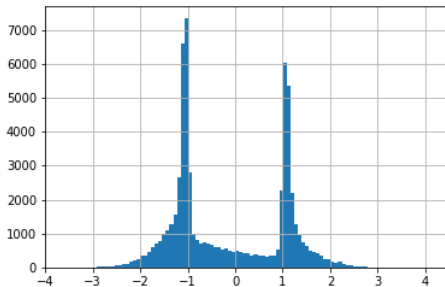
1. Use nltk to remove stopwords from tweets.
2. Tweets are vectorized using several schemes (used ~15K tweets).
3. Best performing scheme is selected for fine tuning.
4. End result is Stochastic Gradient Decent Model with 0.8482 score on large selection of tweets (~64K tweets).
5. Empirical CDF of errors in large test used to estimate probability of making an error.

Vectorized Tweets

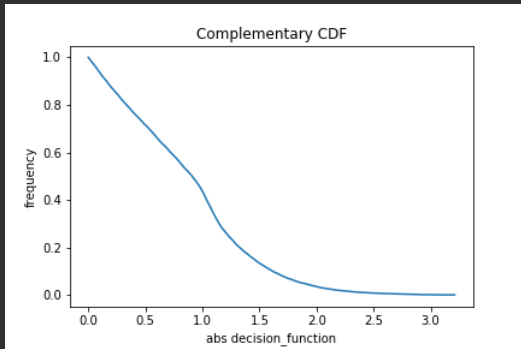
(0, 599005) 0.13724572483536815
(0, 598951) 0.10656565688654346
(0, 597099) 0.07127021892848737
(0, 596894) 0.0696688776808445
(0, 596846) 0.0835728977646812
(0, 596744) 0.07270567427770662
(0, 581647) 0.08857473689547032
(0, 581646) 0.08857473689547032
(0, 568348) 0.21930758305074324
(0, 568331) 0.1444337225102922
(0, 548752) 0.08863406696726873

⋮

Test for normality failed - p-value near zero (couldn't use prediction interval)



Empirical Complementary CDF



Passing user input to flask app

```
1 <div class="col">
2   <h2 class="headers">Party Predictor</h2>
3   <p>
4     Type in a tweet to get a prediction of tweeters political party.
5   </p>
6   <form method="POST">
7     <input name="text">
8     <input type="submit">
9   </form>
10   {{party_prediction}}
11 </div>
```


Second model attempted

Attempted to use Stochastic Gradient Decent Regression Model to get dw nominate score of tweeter.

Training $R^2 = 0.95$ and test $R^2 = 0.5$.

Heroku deployment features

- About page - interactive plots.
- ML models page - text box to predict party of tweeter.
- Search Media Scores Page - text box to search SQL data base for media domain.

Heroku difficulty

Flask app, `app.py`, is in a separate directory from heroku config files. This made it difficult to load the scikit learn models into the app. The issue was resolved using by printing information to the page on the current working directory using the `os` library if the models failed to run:

```
1  if loading_error:
2      def list_tostring(input_list):
3          return ' '.join(input_list)
4      party_result = os.getcwd()+' loaded '+str(num_loaded)+' '+list_tostring(os.listdir())
5  else:
```

SQL Query

```
1 select domain_name, domain_count, nominate_dim1, nominate_dim2, dim1_stddev,  
2 dim2_stddev, round((dim1_stddev/sqrt(domain_count-1))::numeric,3) as dim1_stderr,  
3 round((dim2_stddev/sqrt(domain_count-1))::numeric,3) as dim2_stderr  
4 into final_db  
5 from tenthousand_db  
6 where domain_count>1  
7 group by domain_name, domain_count, nominate_dim1, nominate_dim2, dim1_stddev, dim2_stddev  
8 order by domain_count desc
```

jsonToCSV

```
1 def jsonToCSV(json_path, csv_path):
2     import json
3     import csv
4     merged_csv = []
5     with open(json_path, encoding='utf-8') as ref:
6         data = json.load(ref)
7         headers = list(data[0].keys())
8         csv_row = []
9         for item in data:
10             item_ls = []
11             for col in headers:
12                 try:
13                     item_ls.append(item[col])
14                 except:
15                     item_ls.append(None)
16             csv_row.append(item_ls)
17         merged_csv += csv_row
18
19     with open(csv_path, 'w', newline='', encoding='utf-8') as csvfile:
20         spamwriter = csv.writer(csvfile)
21         for row in merged_csv:
22             spamwriter.writerow(row)
```

Questions