Title

Analysis of semantic bias and sentiment across major American news outlets.

Problem

The core motivation for this project is to address the increasing divergence in presentation of the same events and issues in the media. Empirically, we see a large spectrum of “spin” applied to news articles, which varies from slight bias in word choice to completely misrepresenting facts, depending on the news outlet and topic. Even if the average individual gathers their news from multiple sources, it can be challenging to discern which, if any, source is representing it accurately. In cases where individuals draw from only one source, it’s critical to understand its biases. Using Natural Language Processing, my goal is to provide average folks a tool to analyze news articles as they consume them and raise awareness of “spin” in the article.

Methodology

The training data I’ve selected (https://www.kaggle.com/snapcrack/all-the-news) is composed of ~140k articles, from 15 major U.S. news sources, with articles dating roughly from 2013 to 2017. The initial data format is as simple as possible, mimicking what you could quickly scrape from a news site, and includes basic columns like title, publication, author, date, and article text. This data is not tagged in the supervised learning sense, i.e. it is not known from the dataset the level of bias in a given article.

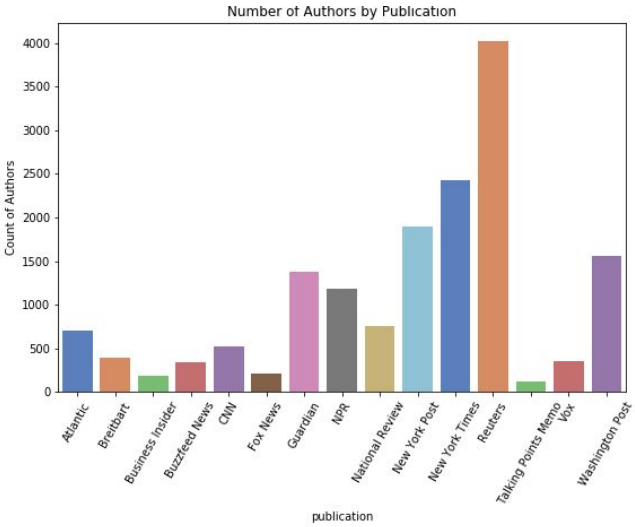
The general methodology is as follows:

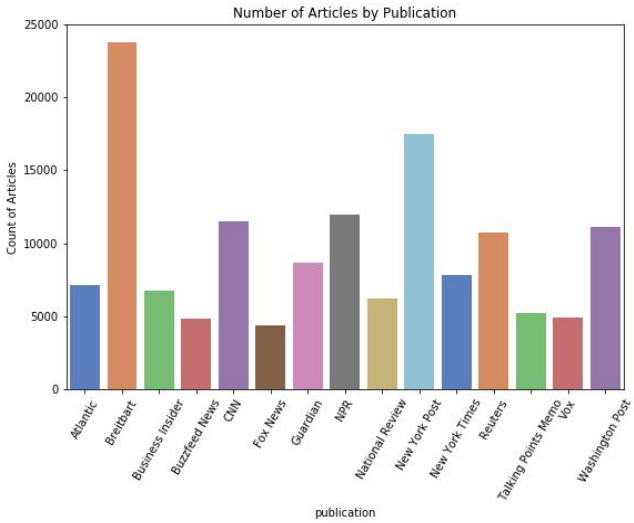
1. Data Wrangling
    a. Process raw text by removing punctuation and stopwords, tokenizing the cleaned text, and finally lemmatizing the tokens for better matching against existing corpora. *(nltk)*
    b. Convert clean text to a tf-idf vector array, to account for varying lengths of articles. *(sklearn)*
    c. (Poor Results, not in production) Apply SVD of resulting feature matrix to reduce dimensionality. This feature is was removed due to low **explained variance.** *(sklearn)*
2. Topic Modeling
    a. Use an efficient unsupervised clustering algorithm (KMeans) to group similar articles into topics by word prevalence (tf-idf). *(sklearn)*
    b. Tune number of clusters to minimize inertia.
    c. Apply LDA to determine top 5 tokens (keywords) for each topic (cluster). *(sklearn)*
3. Sentiment / Subjectivity Analysis
    a. Update corpora of VaderSentiment model with MPQA Opinion corpus, which is human-scored corpus of ~1500 news articles and related content, to improve accuracy against the training set *(vaderSentiment)*

b. In order to provide bias analysis, or subjectivity analysis, as well as sentiment analysis for this **unsupervised** learning problem, it is necessary to use a corpus or dictionary already tagged with polarity scores. I utilized the vaderSentiment library, which is a sentiment analysis model trained on a variety of corpora, including 500 New York Times articles, movie reviews, and Tweets. Once transformed, the model returned a matrix of both sentiment (pos/neg) and neutrality, which was used to derive subjectivity (bias) for each article.
c. Apply trained model to raw text and parse out positive, negative and neutral scores.

Exploratory Data Analysis

Skewing:
I was initially concerned that the dataset would be dominated by a couple publications, however, the distribution of articles is fairly balanced. An interesting factor, that may tangentially affect aggregate subjectivity analysis of each source, is the wide variance in number of unique authors by news source, with Reuters naturally sourcing articles from many more journalists than others.





Pre-processing

Cleaning: Removed all punctuation and English stop words.
Tokenization: Lemmatized article text and tokenized in preparation for vectorization.
Vectorization: Given variable article length, I used Tfidf to vectorized the article text tokens. The result is a sparse matrix (140k x ~200k) containing only numeric data that be passed to a variety of algorithms.

Topic Modeling

Topic modeling for this unlabeled dataset is an unsupervised clustering problem. There are quite a few algorithms in this space, however, the dataset itself present a constraint in its sheer size. After processing, the feature space is 140k observations by 200k features, where each

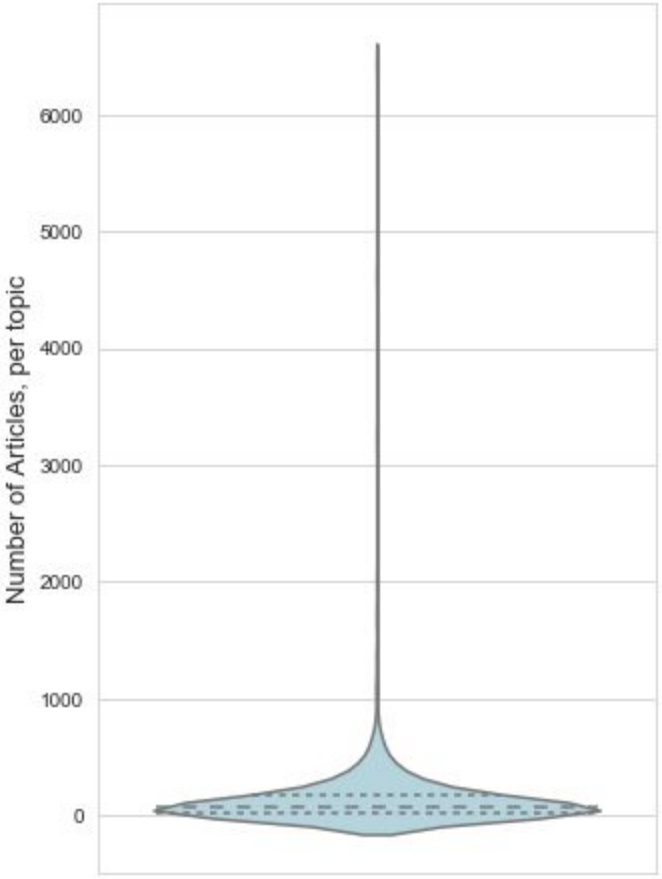
feature is a boolean for whether a given word is found in the article. One of the first challenges was determining how many clusters (i.e. topics) to use. I experimented with a couple algorithms where number of clusters is determined by the algorithm itself and does not need to be passed in as a hyperparameter, such as HDBSCAN (a supposedly more efficient implementation of DBSCAN), however, most could not complete training due to memory issues, even on EC2 instances optimized for large memory computations.

Even using KMeans, one of the most efficient clustering algorithms, I could only train up to 50 clusters (given that time complexity for KMeans is O(n * K * I * d) where n : number of points, K : number of clusters, I : number of iterations, and d : number of attributes). Clearly a more streamlined approach was necessary.

My first solution was to reduce dimensionality using singular value decomposition (SVD) to reduce the dimensionality, and thus size, of the training set. However, using SVD with a higher number of features (~100k, 50% reduction) yielded similar performance issues. With a lower number of features (n=1000), and reduced time complexity, the explained variance by the reduced feature set was far too low to be useful (2%).

Finally, I utilized MiniBatch KMeans, which uses large subsets of the whole data set to produce centroids, then converges based on the same optimization function as KMeans. There may be some small sacrifice in accuracy using this approach, but for this project the ability to run iterations with many different *k* values is more critical. Using MBKM, I applied the Elbow Method against *inertia* to determine the optimal number of clusters, between 50 and 5000, resulting in a local minima where k-clusters = 1000. After training the model with k=1000, I merged in the topic_ids into the original dataframe.

Once I could subset the data into topics, I used LDA to reduce the texts of all articles within each topic to the top 5 keywords, which was helpful in spot-checking topic assignment. For example, the article with title “'Can Carbon Capture Technology Prosper Under Trump?” was assigned to the topic with keywords: climate, carbon, change, tax, trump. This approach was far from perfect as a seemingly similar article, “Weak Federal Powers Could Limit Trump’s Climate-Policy Rollback“ fell into the topic with keywords: trump, donald, clinton, hillary, say. As adding in topic keywords is not critical to this project, I did not proceed with improvements such as adding n-grams or named entity recognition to clean up the keyword categories.



In observing the distribution of articles across the topic_ids, an IQR of 172 to 21 articles makes for fairly even distribution, though there are some outlier clusters with very low and very high numbers of articles (1 and 6448 respectively). Anecdotally, the number of articles per topic seems too high for each topic to pertain to a single event or issue, but rather likely refer to a broader theme, like overall climate change policies under an administration, rather than passing of a piece of legislation like Paris Climate Accord.

The nature of the unsupervised clustering problem is such that there is no final accuracy metric like precision or recall, as in a supervised classification problem, so it's impossible to judge performance by in the sense of "85% of the articles were classified into the correct topic". Working with fit metrics like inertia are helpful in determining the order of magnitude for how many topic clusters to create, and generally it appeared to perform well with spot-checking articles in given topics, however, there were clear exceptions. Looking at the distribution of number of articles written per topic also supports this conclusion, as most topics had a believable number of articles assign, but as *n* increases, the topic must either a be a huge, world-changing event, or sub-topics combined in error. Though the performance was a constraint to using a non-parameterized model, like HDBSCAN, to determine optimal clusters, in the future, I'd like to revisit that approach.

Subjectivity and Sentiment Analysis

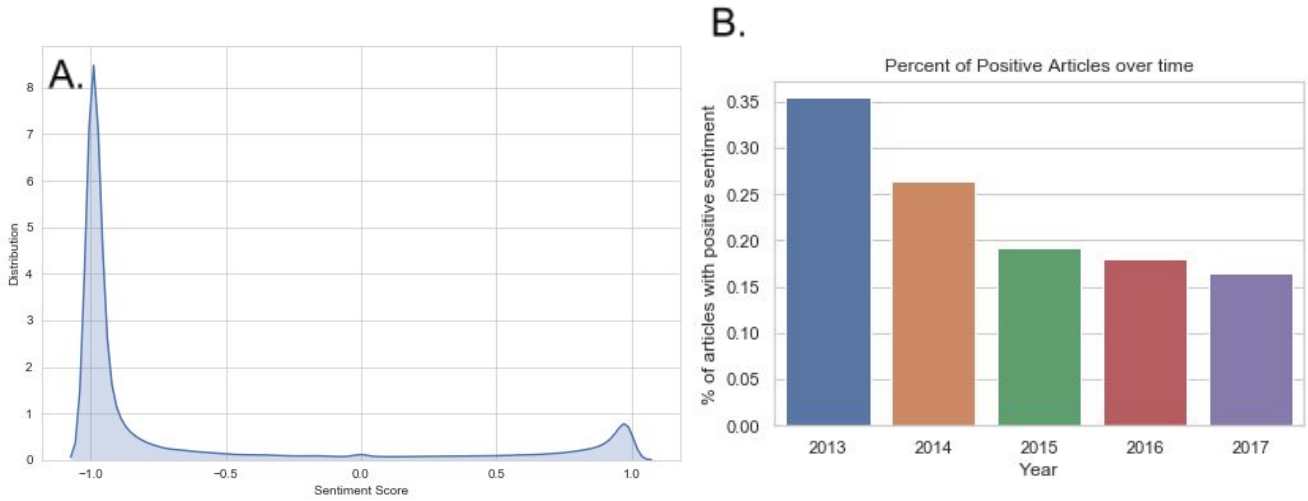
As mentioned above, both subjectivity and sentiment analysis was performed using the vaderSentiment library, which was chosen for its variety of tagged texts in its corpora, including one set of 500 articles from the New York Times. My suspicion was that the accuracy of the sentiment/subjectivity analysis would depend greatly on how similar the corpus is to the project data. In order to improve the corpus further, I added in the MPQA Opinion corpus, which contains ~1500 human-annotated texts from a variety of news sources.

With regard to term definitions, sentiment compares the prevalence and polarity of positive and negative words in a text, arriving at a compound score, with >0 being positive and <0 being negative. Subjectivity compares the ratio of positive/negative words to neutral words in a text, with 0 being perfectly neutral and 1 being 100% subjective. Combining both data points into a matrix is critical here as it's possible to have an article with a polar sentiment, e.g. many negative terms when reporting on a plane crash, that is either non-biased or strongly subjective.

After completing the above steps, the dataset contains in addition to basic columns like article text, title, publication, etc:

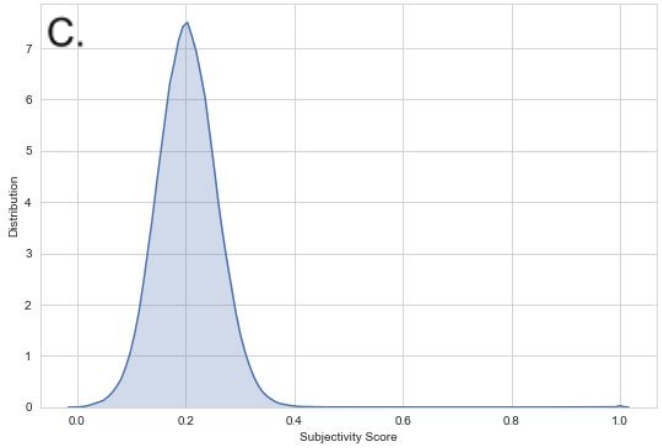
- Sentiment score
- Subjectivity score
- Topic_id
- Topic keywords

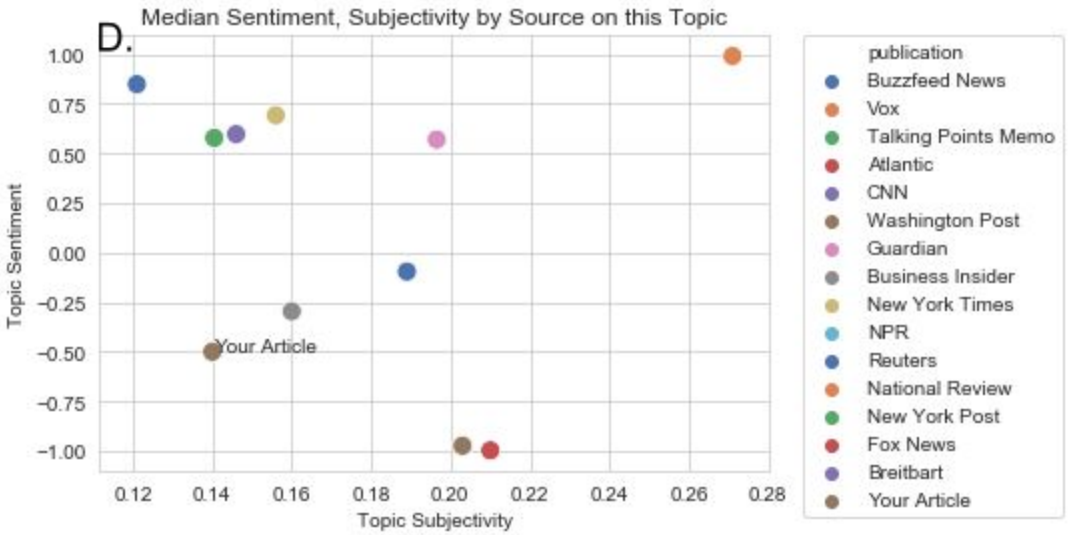
With these additional data points, we can look at the data in aggregate, the results of which are not unexpected. For example, news sentiment is bimodal, with concentration at very positive or



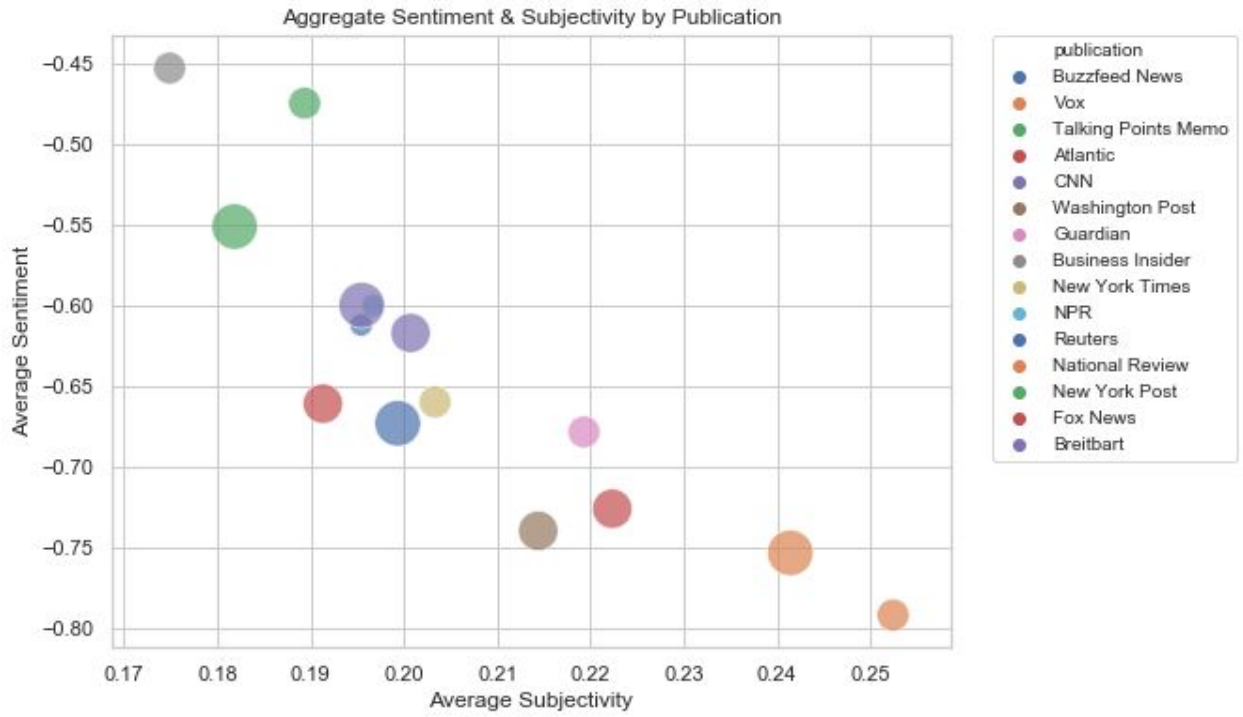
very negative, though there are significantly more negative articles than positive (82% are negative) (A). The proportion of positive articles is also decreasing rapidly over time (B). Subjectivity is fairly normally distributed around .2 (slightly subjective) (C). When analyzing a

single article within a topic, the average sentiment and subjectivity of other new sources for other articles on the same topic can be presented for context (D).





I'd expected to see a clear pattern of high subjectivity and polar sentiment at the ends of the anecdotal bias ranking, either if not in aggregate, then certainly by topic. For example, I'd expect to see Buzzfeed and Breitbart to be more subjective than NPR or Reuters. However, as shown in the chart below, the results are much less conclusive. Whether this challenges my assumptions or the accuracy of the model is not determined.



Sentiment appears to be correlated with subjectivity, when taken in aggregate by news source - the more negative a source is in sentiment of its articles, the more subjective it tends to be. Bubble size reflects the anecdotal bias ranking, based on a consensus of research of several online sources (e.g. here). If these rankings and the analysis methods used in this project were highly accurate, we'd expect to see smaller bubbles in the upper left quadrant and larger in the lower right.