DS5110 - Fall 2019

ASSESSING SIMILARITIES AND DIFFERENCES BETWEEN NEWS ORGANISATIONS IN THE UNITED STATES

Devanshi Deswal, Samar Dikshit, Connor Higgins, Kartheek Karnati, Oliver Spohngellert

Summary

News sources are meant to provide people with facts summarising an event of significance that has taken place. From these facts, we can form our own opinions related to the event, without being influenced by these organisations. However, over time, these sites have developed biases, especially when it comes to politics [1]. Due to this bias, the same event might be reported on in different manners depending on the political lean of a news site.

In this project, we attempt to investigate the similarities and differences between new organisations reporting on political issues in the United States of America. By analysing attributes such as article headlines, vocabulary used, and the sentiment with which an event is covered, we believe that we can find a correlation between these factors and the political lean of the news site. This will also help in uncovering the differences between organisations on the same side of the political spectrum. Our final goal is to build classifiers that can predict the political lean of an article based on its text.

To keep the objectives of this project relevant, we chose to analyse articles that report ongoing political issues, such as the 2020 Democratic Party Presidential Primaries, and the Trump - Ukraine Scandal. As there is no dataset available that fits our specific requirement, we scraped various news sites from all sides of the political spectrum to create a dataset. After filtering out certain articles, the final dataset contains attributes such as the article text, author, and date of publishing, along with metadata such as the last modified date and article tag.

To achieve our goals, we performed a variety of analyses, including word association analysis, reading level analysis, and creating two classifiers. Using graphical visualisations, we could see clear differences in the way events were reported, both over time and by the political lean of the media organisation. Our classifiers were able to predict the lean of an article with over 70% accuracy

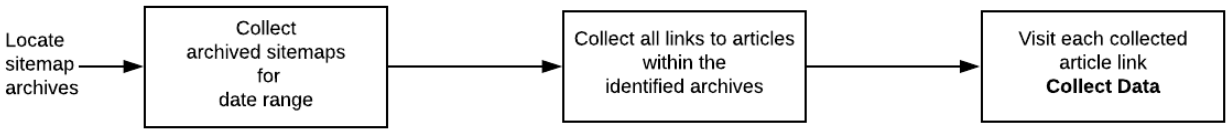
Methods

1. Data Collection

To acquire our data we used a collection of web crawlers written in both R and Python. The final breakdown amongst the data collected was as follows:

- R: Fox News (17,052 articles), Reuters (21,222 articles), Mother Jones (756 articles), BBC (3,377 articles), FiveThirtyEight (523 articles)
- Python: Breitbart (9,260 articles), Vox (7,958 articles)
- Mix: CNN (13,070 articles) -- originally collected in R, bugfix later written in Python

All web crawlers, both written in Python and R followed the same general process:



That is, all of our web crawlers used pre-existing site archives for easy data collection. These "site archives" are essentially just old sitemaps that have been kept. Many sites, maintain sitemaps mainly so they can be indexed by search engine web crawlers [3]. For news sites, showing up in search engines is vital and so we could reliably count on most sites being exceptionally easy to scrape. As an example, https://fivethirtyeight.com/sitemap-index-1.xml.

Sitemaps might be meant for search engines, but they are very easy to use by crawlers we write as well. This meant simple, fast code could be used to assemble a very large dataset with minimal effort. This had the added advantage of being an extremely modular approach that was easy to parallelize.

We laid out a few basic rules for data collection. First we only looked for articles running back since August 2019. The primary reason for this was to ensure we could maintain a balanced dataset. Some sites, such as FiveThirtyEight were very well maintained and collection from the beginning of publication was very simple. Others, such as Reuters only kept articles from the last few months. To reconcile these differences, we picked a date range that let us account for them and avoid creating artificial imbalances (i.e. were Reuters would appear to "write less" than other sites past a certain date).

In addition we wanted to limit the topic strictly to US politics, as classification of political bias (primarily in the US) was the main project goal. As much of the time the topic of an article was only available *after* collecting data from it, it became necessary to establish a rigorous filtering process to maintain data quality.

2. Data Filtering

While the web scrapers collected 72,218 articles from 8 different news sites, most of these articles were not related to US politics. This is because most news websites archive their articles based on the date that they were published on, and not by topic. For this reason, we created a pipeline to filter out irrelevant articles and keep what was required for our analysis.

The stages of the pipeline were:

1. Use article metadata to get articles related to US politics: All the news sites we collected data from use an article tag to classify the topic of the article. For example, CNN uses the tag *politics* for articles related to US politics. However, not all news sites use the same tags, for example Mother Jones uses a variety of tags such as *Politics*, *Impeachment*, and *Donald Trump*. Hence we first had to identify tags pertinent to our topic to ensure that we get as many relevant articles as possible.
2. Check if the article text is not *NA*: R uses *NA* to represent missing data. Some news sites publish podcast videos on their websites, which can result in the scraper collecting an article with relevant metadata but no article text. As we're performing analysis on the text and using it as a feature for our classifiers, we need to remove any data collected where it text is missing, i.e. *NA*.
3. Transform data by adding labels: To classify articles by political lean and source, we added the two attributes as labels for each observation. Using the media bias ratings of AllSides [1], we categorised the political lean of a certain source's article as *left*, *right*, or *centre*. If AllSides classified a source as leaning left/right, we classified it as left/right. The second label we added was just the source of the article, for example *fox* for Fox News, to keep our news site labels consistent across the data.
4. Apply date filter: To ensure that we're analysing events that have either just taken place or are currently ongoing, all the articles in our final dataset were published on or after 1st August 2019. Articles published before this date were removed.

All stages of the pipeline were implemented using functions provided in R's *dplyr* package. The final dataset we obtained had 8,992 observations, with 13 attributes for each article.

3. Data Analysis

3.1 The Most Common Terms

When looking at articles, one could argue that the text of the headline is more important than the text of the article itself, as everyone reads the headline, but not everyone reads the article. Further, headlines have a lot of data on what topics different news organizations are reporting on. In addition, we wanted to see how article headlines changed with time on both sides of the political spectrum. With these motivations in mind, we decided to analyze the top words used in article headlines in both right and left leaning news organizations. To do this, the *article_headline* column was split into its individual words using *unnest_tokens()*. Then, stop words were filtered out, and the remaining words were grouped by their *political_lean* and *month*, counted using *count()*, and filtered to the top 10 words using *topn()*. This was plotted using *geom_col()*, and faceted using *facet_wrap()* by month. This allowed us to see the most common words in headlines by both right and left wing news sources.

3.2 Presidential Candidate Popularity Analysis

From the data that we have, we were interested to see the coverage each person running for president in the next election was getting from each news site. In order to see if there was a disparity in coverage for the candidates, we plotted a bar graph for the 5 most covered people on each site running for President faceted by site and filled by lean. In order to get the intended output, the data in the *article_text* column of the

dataset was split into individual words with the *unnest_tokens()* function from the *tokenizers* package in R grouped by news site and political lean using *group_by()* function from the *dplyr* package. The resulting data frame was called *tidy_articles*. Stop words were removed using *anti_join()* function. We also created a tibble called *names* of last names of candidates running for President in the next election. We performed a semi join on the data frame which had individual words of the article text with the aforementioned tibble and visualized the top 5 most covered people running for President using *top_n(), ggplot(),* and *geom_col()* functions.

3.3 Bigram and Word Association Analysis

In order to view how different new organisations use different terminology, we analysed the most common bigrams in article text for each political lean. To visualise this, we used the *igraph* and *ggraph* packages in R. In order to obtain bigrams from text data, we used the *unnest_tokens()* with token set to *ngrams*, and *n* equal to 2. After getting the bigrams, we split them into individual words (*word1* and *word2*), and filtered out stopwords using *separate()* and *anti_join()* respectively. We also kept only the necessary columns in the dataset i.e. *date*, *news site*, *political lean*, *word1* and *word2*.

On the resultant data frame, we used *filter()* to get left leaning articles, and got the counts of bigrams appearing more than 200 times. Using *graph_from_data_frame()* and *ggraph(),* we were able to visualise the bigrams. Similarly, we did the same for bigrams appearing right leaning articles, and for bigrams appearing the most in the months of August and September 2019.

During exploratory data analysis, we found two date ranges where the total number of articles, regardless of political lean peaked, and chose to analyse these articles further. These date ranges were 15th to 17th October, and 5th to 7th November. For these dates, we analysed bigrams appearing in article headlines. The bigrams were obtained and visualised in a similar fashion to what was performed for the article text analysis.

3.4 Article Reading Level Analysis

One interesting, and politically relevant, facet of the problem to explore is the "sophistication" of each news organization's writing style. To do so we applied the Flesch-Kincaid readability scale to the collected news article data. Developed for the Navy by Flesch Kincaid in 1975 [2], it provides a simple formulaic approximation of the years of schooling needed to understand a given text based on the number of words per sentence and number of syllables.

3.5 Sentiments and Word Clouds

To extract the subjective information from the text we classified the polarity in the words that make up the text. We used the bing lexicon to predict the sentiment score(positive – negative) and analyze if the overall underlying sentiment is positive or negative. To perform the analysis, we tokenized data to bigrams using *unnest_tokens()* function, specifying token value as ngram with n=2. With antijoin of negation words with token in column 1 we eliminated the probability of wrong categorization by taking the reference into consideration. To examine how these sentiments changed over time we plotted a box plot over the period of time to validate the uniformity/ variation in the trend of the writing style grouping by both news sources and political lean. To further evaluate the strength of each sentiment for a word we performed a similar analysis on our tokenized data using the afinn lexicon which assigned scores for positive and negative sentiments followed by likewise plotting a box plot using *ggplot()* function.

We used *wordcloud()* function to finally present the keywords standing out in the text through a visual engagement. These words were expected to depict the change/uniformity in the observation observed in the barplots for bing and afinn lexicons. To create the word cloud we loaded the required packages including the ones for mining the text, stemming it, generating the word cloud and for selecting the colors from the color palettes. The *Corpus()* function was then used to load the data. Once the data was loaded it was then transformed to a tidy format which involved removing stop words, pronunciation, special characters and also popular names. With the help of *TermDocumentMatrix()* function we obtain a table containing the frequency of word which was used as an input to the *wordcloud()* function for generating the word cloud.

4. Article Classifier

4.1 Support Vector Machine Classifier

One of our objectives was to see how accurately would a machine learning algorithm differentiate a right leaning article from a left leaning one. To achieve this goal, we fit a Support Vector Machine (SVM) classification model to our data excluding centre leaning articles.

We first created a dataframe of individual words from the article text using *unnest_tokens()* function and filtered out the stop words and all words that appeared less than 10 times in all the articles using *count()* and *filter()* respectively, and named it *tiar*. Similarly, we got another data frame with an additional column stating which article that word belonged to, and the lean of the article. We inner joined *tiar* with this data frame, and then cast it into a document-term matrix using *cast_dtm()* and named it *tilear*. We then coerced *tilear* into a matrix and created a partition on it to form a training and a test set with 0.8 and 0.2 proportions of the data respectively. The train and test sets were then coerced into data frames using *as.data.frame()*. We then removed the *lean* column on both data frames to pass them into the classifier. The training set was then passed to the *train()* function of the *caret* package in R using the method *svmLinearWeights2* to train the classifier. Once the training was done, we predicted the accuracy on the test set and obtained the confusion matrix along with diagnostics such as specificity and sensitivity (here, left is the positive class and gets classified as 0).

4.2 Fasttext Classifier

Fasttext is a classifier system developed by Facebook which creates word embeddings in order to capture the meaning of words. It does so by going through “ngrams” of words, meaning it learns the meaning of words by learning the meaning of subsections of words. This helps the algorithm to learn prefixes and suffixes.

In order to use fasttext, the data was reorganized into the format that the fasttext command line tool accepts, where each line starts with __label__<label of text> followed by the text itself. The fasttext command line tool was then run using k-fold cross validation to find the best hyperparameters. The tool was then used to make predictions on the test set, and the validity of these predictions were analyzed in python, including generating an ROC curve.

4.3 RNN Classifier

Another classification method that was used was a RNN. RNNs generally are often used in natural language processing applications, as they are able to use the sequence of information as a factor in the classification process very effectively. The RNN we trained was implemented in Keras.

Before the text was used in the model, it was processed to be converted to pretrained embeddings on the entire English language from Fasttext. Only the first 200 words of each article were used in order to decrease training time. Once this data was generated, it was first fed into a *Conv1D* unit with *MaxPooling* to decrease data dimensionality and learn general patterns. This was then fed to a 128 size *LSTM* unit, then to a traditional feed forward neural network where the predicted label was output. See Appendix B, Section 1 for code samples.

Results

1. The Most Common Terms: We found that the word 'Trump' was by far the most used word in headlines for both right and left wing news outlets every month. This is unsurprising, as Donald Trump is the President of the United States, and a controversial one at that. The word 'impeachment' was not one of the top 10 words in August, but picked up in usage every month, peaking in November. In addition, 'impeachment' was used much more in headlines by left wing news outlets, indicating that they are reporting more on impeachment than right wing news outlets. The right used the word 'biden' much more during every month analyzed. This is likely partially due to the fact that they are reporting on Hunter Biden much more than the left, and due to the fact that he is currently the most likely candidate to challenge Donald Trump in the 2020 election.
2. Presidential Candidate Popularity Analysis: We found out that Trump, Biden, Warren, Sanders and Harris got the most coverage from news sites (in that order), apart from 3 instances where BBC covered Andrew Yang more than they did Kamala Harris, Reuters covered Pete Buttigieg more than they did Kamala Harris and Five Thirty Eight covered Joe Biden and Elizabeth Warren more than they did Donald Trump. Another interesting thing we found out was that Tom Steyer received least coverage from all news sites except for FiveThirtyEight and Mother Jones where Andrew Yang received the least coverage.
3. Bigram and Word Association Analysis: We found that left leaning articles had bigrams like 'gun violence', 'mass shootings', 'El Paso', and 'climate change' apart from common bigrams like 'Donald Trump' and 'Volodymyr Zelensky' (the President of Ukraine). What's interesting is that right leaning articles didn't talk about mass shootings and gun violence much, while they mention 'illegal aliens' more often. 'Hillary Clinton' and 'Ocasio Cortez' appeared frequently in right leaning articles. In the month-wise bigram analysis, terms like 'trade war', 'Hong Kong', and 'El Paso' were most common for August due to the ongoing trade war between the US and China, 2019 Hong Kong Protests, and El Paso shooting respectively. Impeachment wasn't used frequently enough to be visible on the visualisation. However, terms like 'impeachment inquiry', 'Volodymyr Zelensky', and 'whistleblower complaint' were used often in articles from September as the Trump-Ukraine Scandal reached public attention during the month. was. During the analysis of article headlines for 15th to 17th October, bigrams related to the Ukraine scandal remained frequently used, along with 'democratic debates' and Democrat presidential candidates such as 'Kamala Harris', 'Elizabeth Warren', and 'Joe Biden' due to a debate on 15th October. Similar to

October, the most common terms between 5th and 7th November were related to the scandal, and included the bigrams ‘Gordon Sondland’ (the US ambassador to the EU), ‘quid pro’, and ‘pro quo’ (which together form the term *quid pro quo*). Another point of note is ‘impeachment trial’ was also used, as opposed to just ‘impeachment inquiry’.

4. Article Reading Level Analysis: This showed some expected relationships, such as BBC (by design) being fairly easy reading while others like Vox are more are more difficult (perhaps due to favoring long-form articles more often).
5. Sentiments and Word Clouds: The left leaning political sites use moderate adjectives and adverbs like ‘important’, ‘wise’, and ‘work’, while the right uses strong words to propel their writing forward, with words like ‘paradoxically’, ‘molestation’, and ‘obliterate’ among the popular ones. CNN, FiveThirtyEight, and Vox (all left leaning) are the only news sources displaying an overall positive sentiment. Meanwhile, Fox displays the most negative sentiment value amongst the remaining five. On grouping the news sources by political lean, we observed that the left displays an overall positive sentiment, the right has an overall negative sentiment, with center lean falling somewhere in the middle. On further analysing it was observed that right leaning news sources integrate words having strong negative sentiments, while the left displayed an overall positive sentiment the positive words did not have a very high sentiment score attached to it and were mostly only moderately positive. From the analysis we could hence conclude that left lean sources write with neutrality keeping their biases aside.
6. Article Classifiers:
    6.1. Fasttext Classifier: The fasttext model achieved 74% accuracy on the test set, including all three classes of left, right, and centre. The model performed best on the class “centre”, with an F1 Score of 0.88, and worse on the class “right”, with an F1 Score of 0.6. Further, as can be seen in Figure 1, the AUC score on centre leaning articles was much better than on those of both left and right leaning articles.
    6.2. SVM Classifier: The results of the SVM classifier were very promising as we got a prediction accuracy of 92.72% on the test set. The sensitivity was 0.9286 and the specificity was 0.9254, i.e. the ratio of correctly predicted left articles to the total number of left articles (true positive rate) was 0.9286 and the ratio of correctly predicted right articles to the total number of right articles (true negative rate) was 0.9254.
    6.3. RNN Classifier: The results of the RNN were excellent, boasting headline accuracy of 0.8771. The classifier performed very well on all classes, with 0.8899 accuracy on left, 0.9149 accuracy on right, and 0.9494 accuracy on centre. Sensitivities and specificities for each class were above 0.8, except for centre’s sensitivity of 0.55. Further, as can be seen from Figure 2, the ROC curves were exceptional for each class, showing that the RNN model has great predictive power.

Discussion

Overall, our results show that there are massive differences in the way that right, left, and centre leaning organizations present the news. Through the use of exploratory data analysis, we identified how different organizations have clear differences in not only their writing styles, but in what they report on. It is clear that, for example, the right is avoiding the topic of impeachment to an extent. In addition, you are much

more likely to find articles about gun violence on left leaning news sites, while you are more likely to find articles about illegal immigration on the right.

In addition to the revelations of our data analysis, our models showed that it is entirely possible to automatically predict the political lean of an article given exclusively the article's text. Both our SVM classifier and our RNN classifier were able to complete this task at a very high level. It could be argued that they performed above human level performance.

Given more time, we would have added much more analysis to the project, as there are many interesting sub areas to look into. For example, it would be interesting to see how these websites reported differently before Trump became president, and during various other big political events, like the Mueller hearings. Further, analyzing on a per author basis could have shown that even within websites different authors have distinct writing styles. Finally, a stretch goal would be to collect white nationalist/extremist content, and analyze the way that it differs from non-extremist content.

Statement of Contributions

1. Devanshi Deswal: Comparison of adjectives and adverbs used by the Left and Right using word clouds, sentiment analysis of article text for each news site and political lean
2. Samar Dikshit: Filtering and transforming the data collected to include articles related to US politics, bigram and word association analysis
3. Connor Higgins: Data collection using web scraping, article reading level analysis using the Flesch-Kincaid scale
4. Kartheek Karnati: Presidential candidate popularity analysis, bigram and word association analysis, SVM article classifier
5. Oliver Spohngellert: Data collection using web scraping, overall term usage analysis, Fasttext article classifier, RNN article classifier

References

[1] "Media Bias Ratings". [Online]. Available: https://www.allsides.com/media-bias/media-bias-ratings
[2] J. P. Kincaid, J. Fishburne, R. R. P., C. R. L., and B. S., "Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel," *Institute for Simulation and Training*, Jan. 1975.
[3] "sitemaps.org," *Home*. [Online]. Available: https://www.sitemaps.org/
[4] "fasttext.cc", *Wiki word vectors*. [Online]. Available: https://fasttext.cc/docs/en/pretrained-vectors.html
[5] "keras.io", *Pretrained word embeddings.* [Online]. Available: https://keras.io/examples/pretrained_word_embeddings/

Appendix A

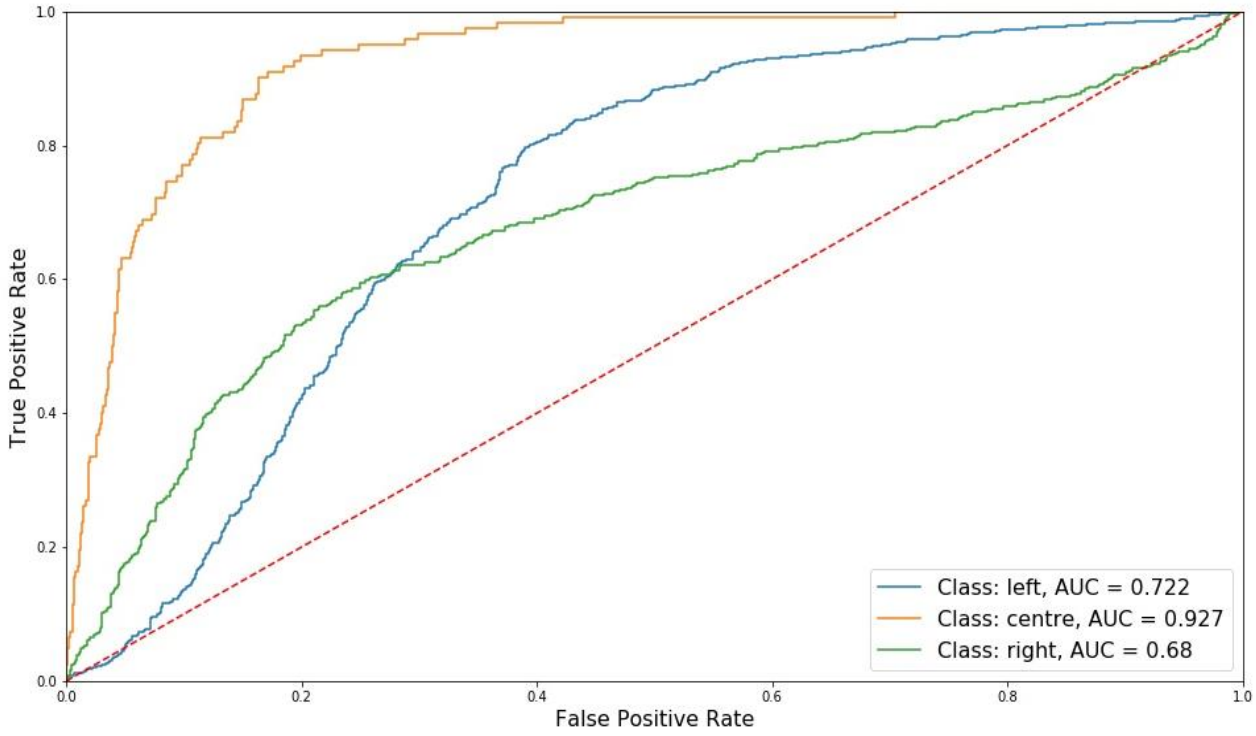


Figure 1: ROC curve for Fasttext classifier

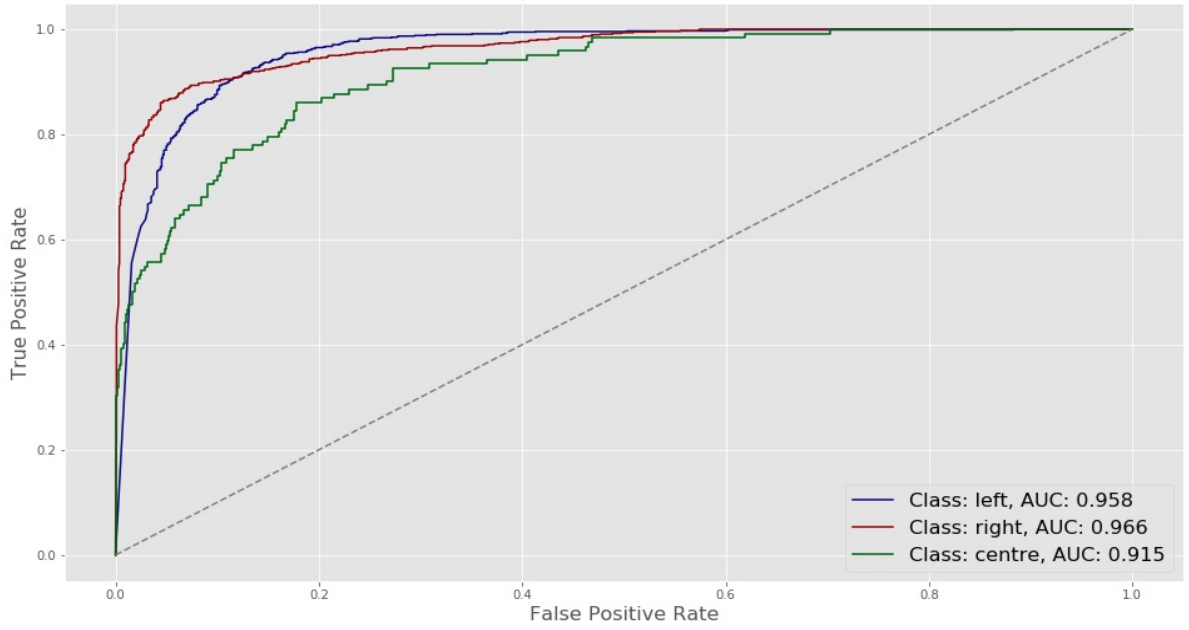


Figure 2: ROC curve for RNN classifier

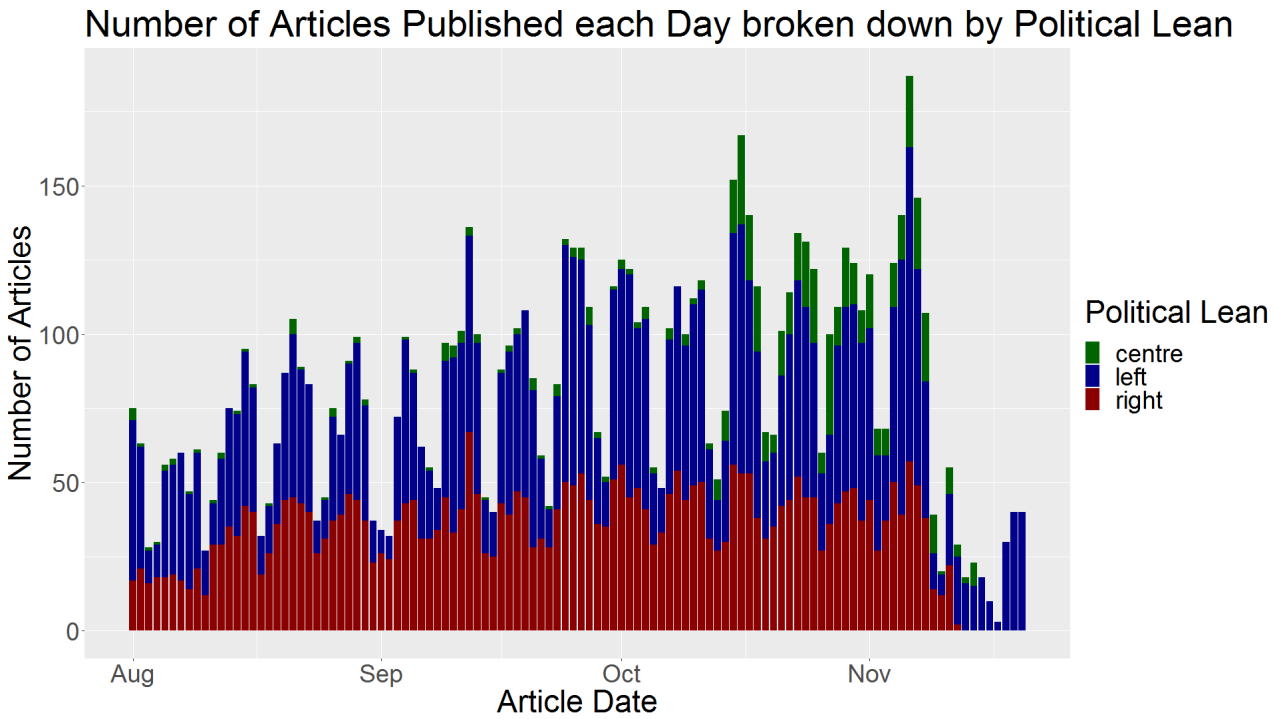


Figure 3: The number of articles published each day broken by political lean

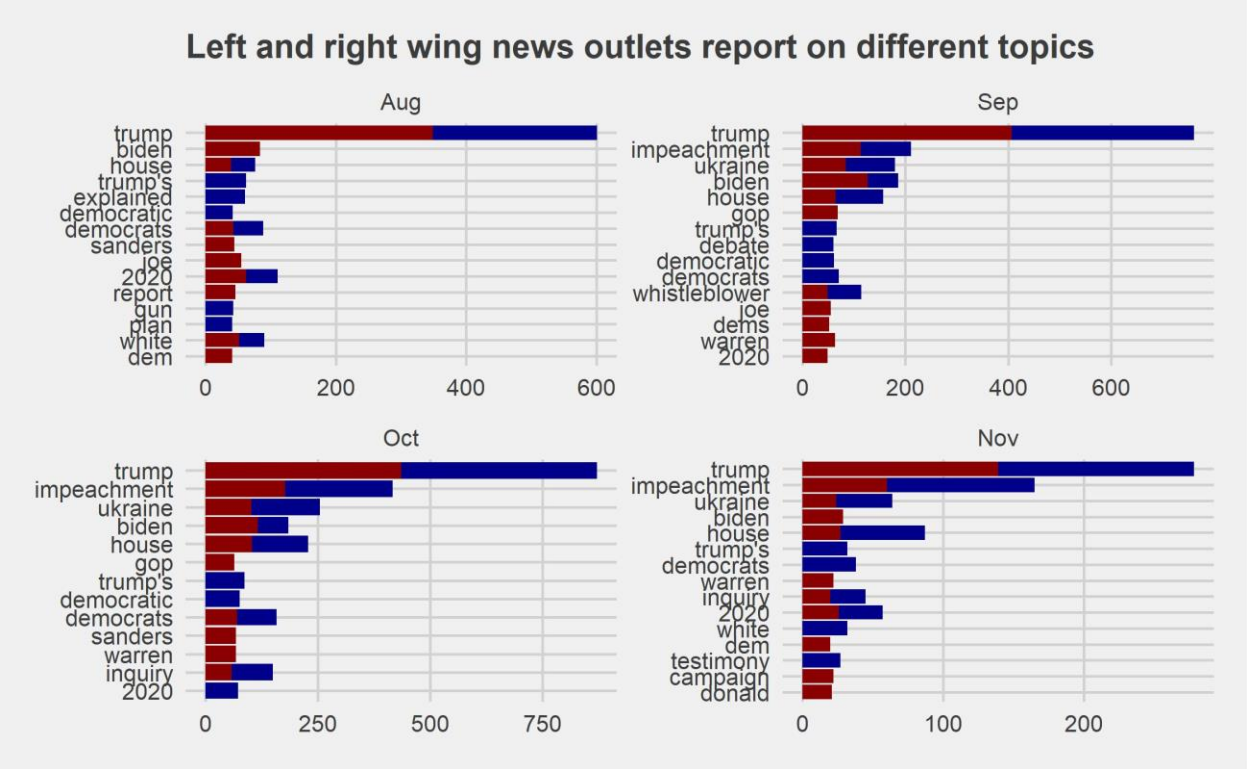


Figure 4: The most common terms

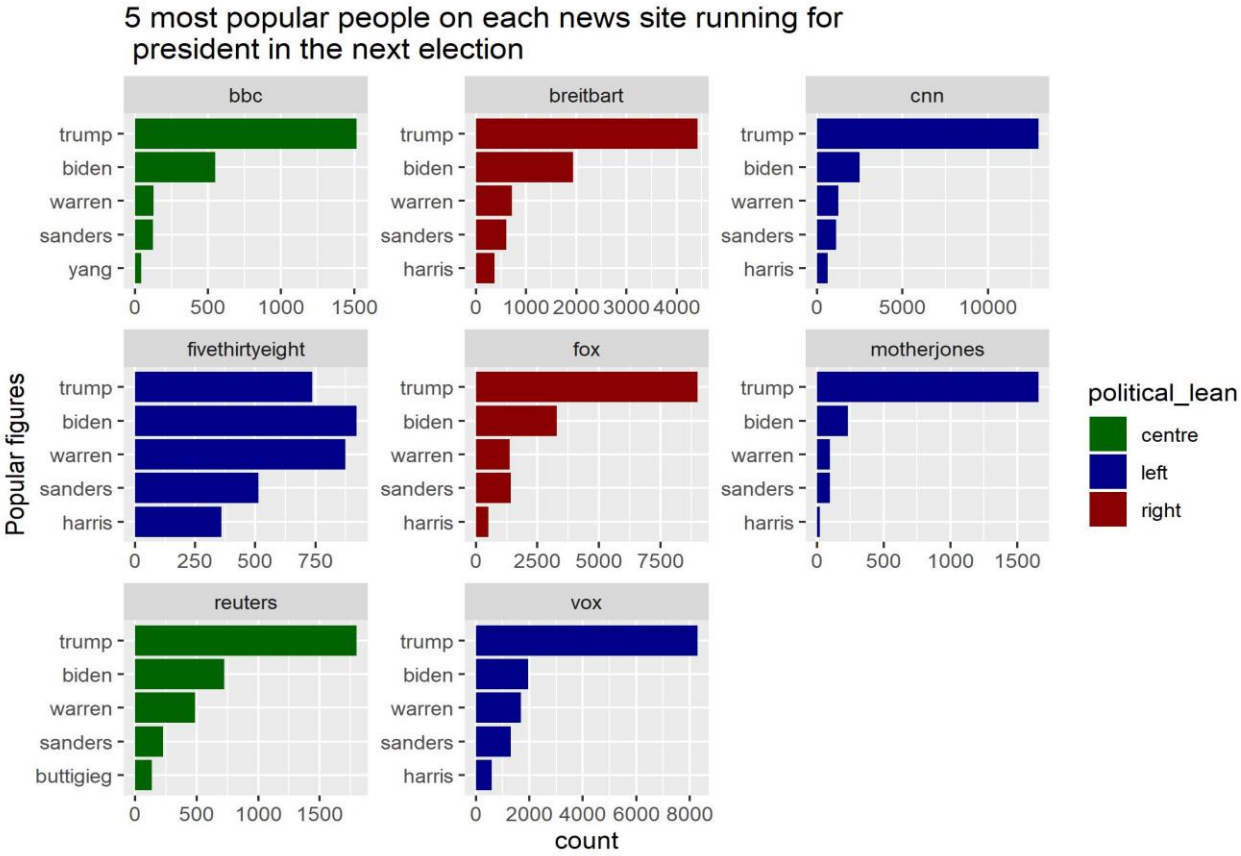


Figure 5: The 5 most popular candidates

```
# A tibble: 8 x 4
# Groups:   news_site, political_lean [8]
  news_site       political_lean word   number_of_mentions
  <chr>           <chr>          <chr>               <int>
1 fox             right          steyer                128
2 fivethirtyeight left           yang                  114
3 vox             left           steyer                112
4 cnn             left           steyer                 84
5 breitbart       right          steyer                 45
6 reuters         centre         steyer                 28
7 bbc             centre         steyer                  7
8 motherjones     left           yang                    4
```

Figure 6: The least popular candidates

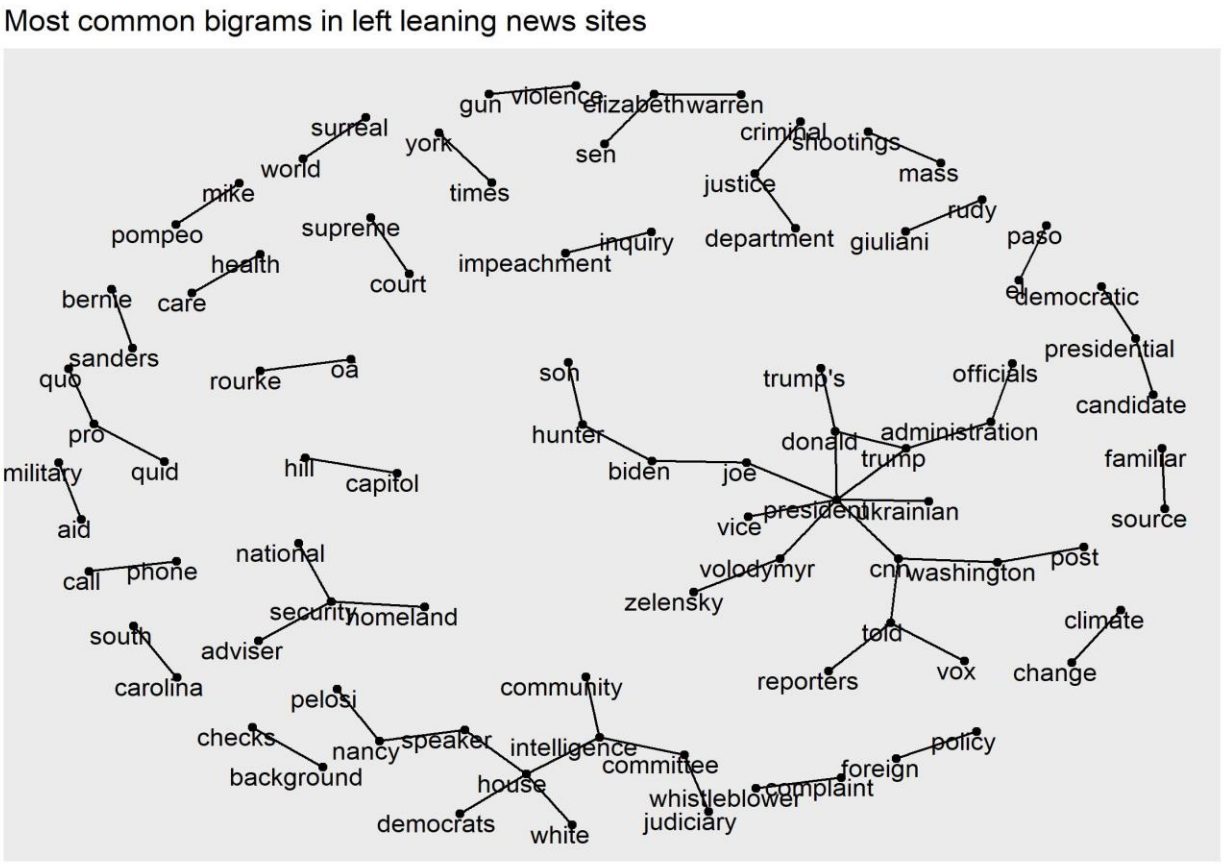


Figure 7: Most common bigrams from left leaning articles

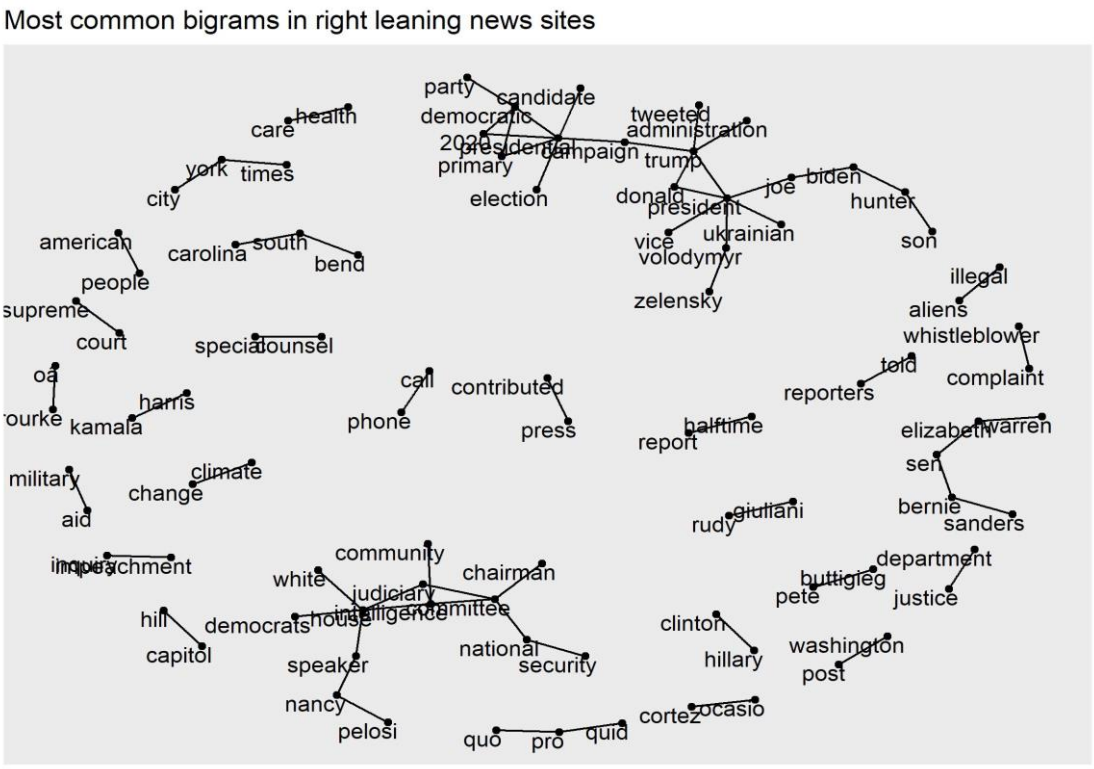


Figure 8: Most common bigrams from right leaning articles

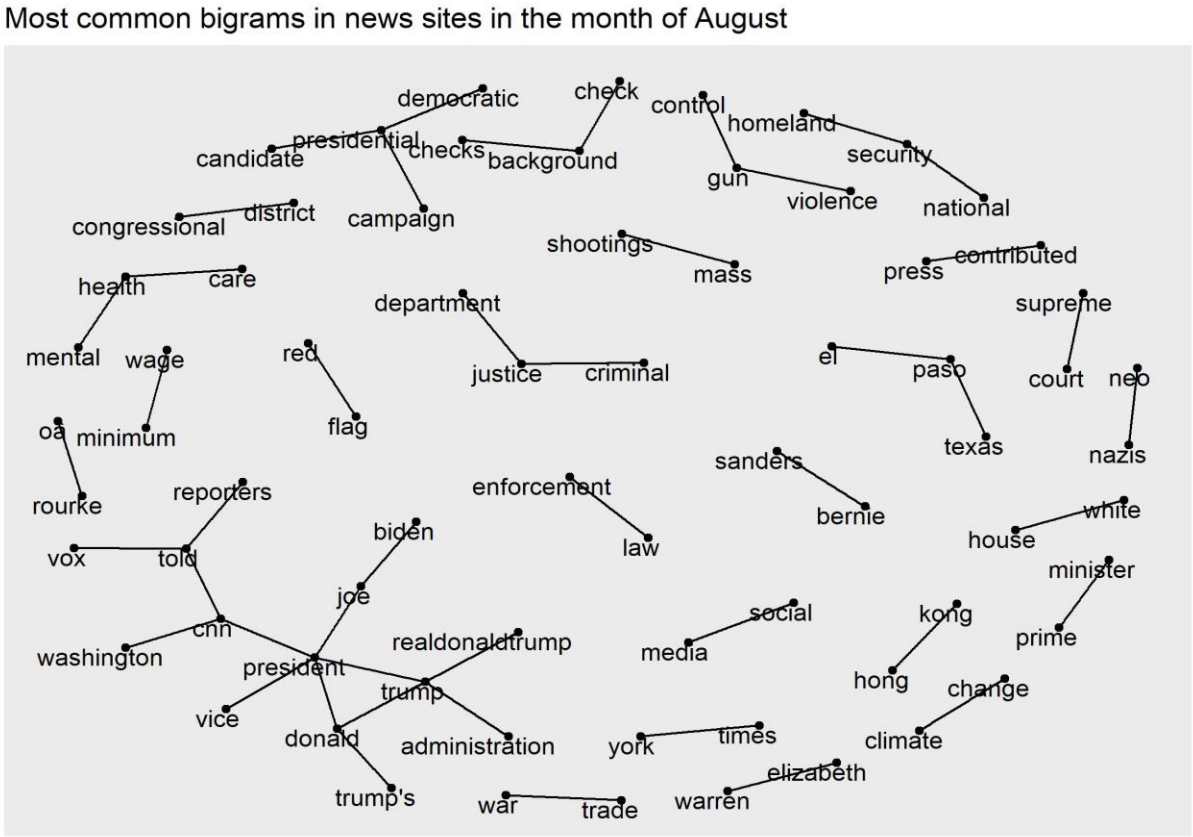


Figure 9: Most common bigrams in August

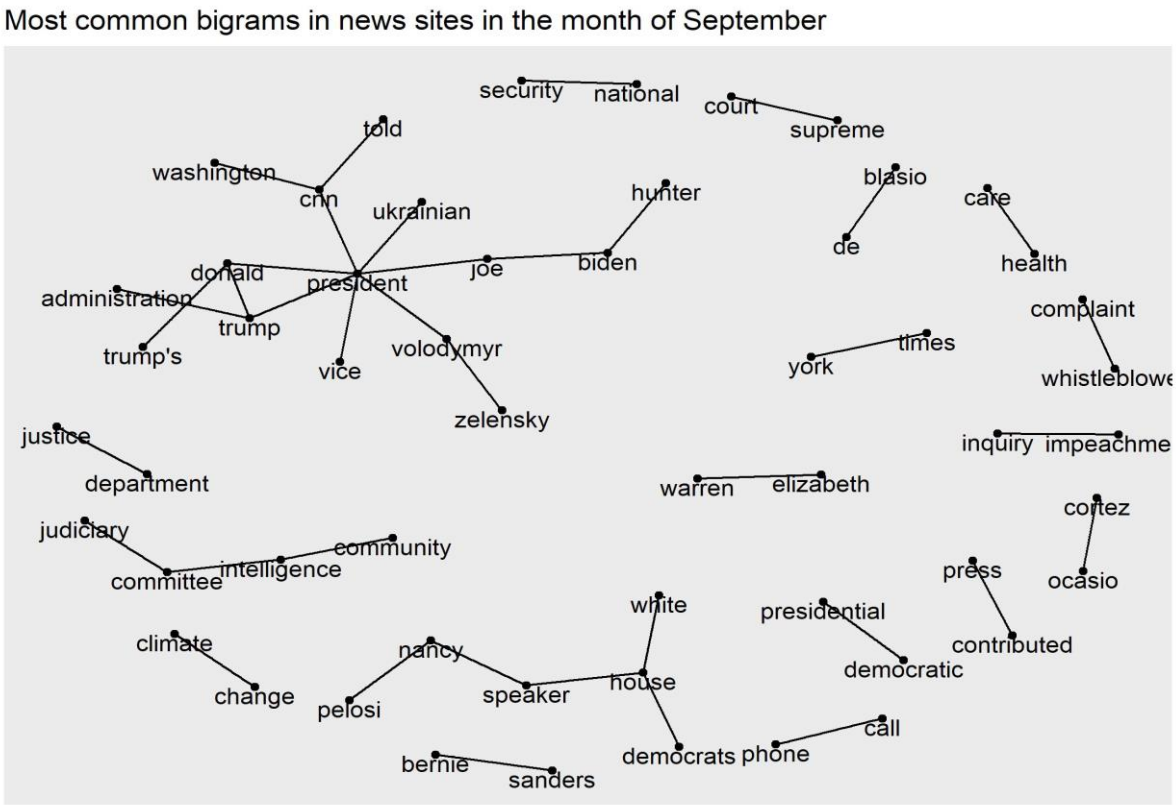


Figure 10: Most common bigrams in September

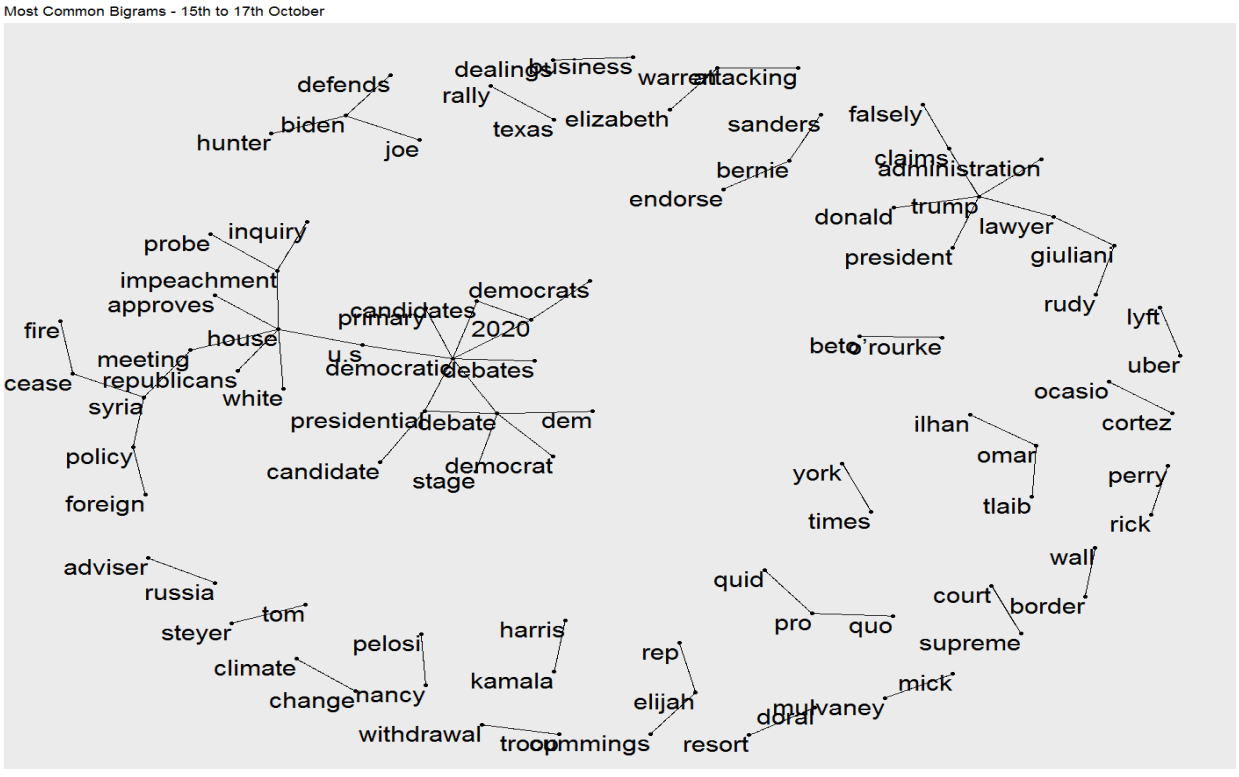


Figure 11: Most common bigrams from 15th to 17th October

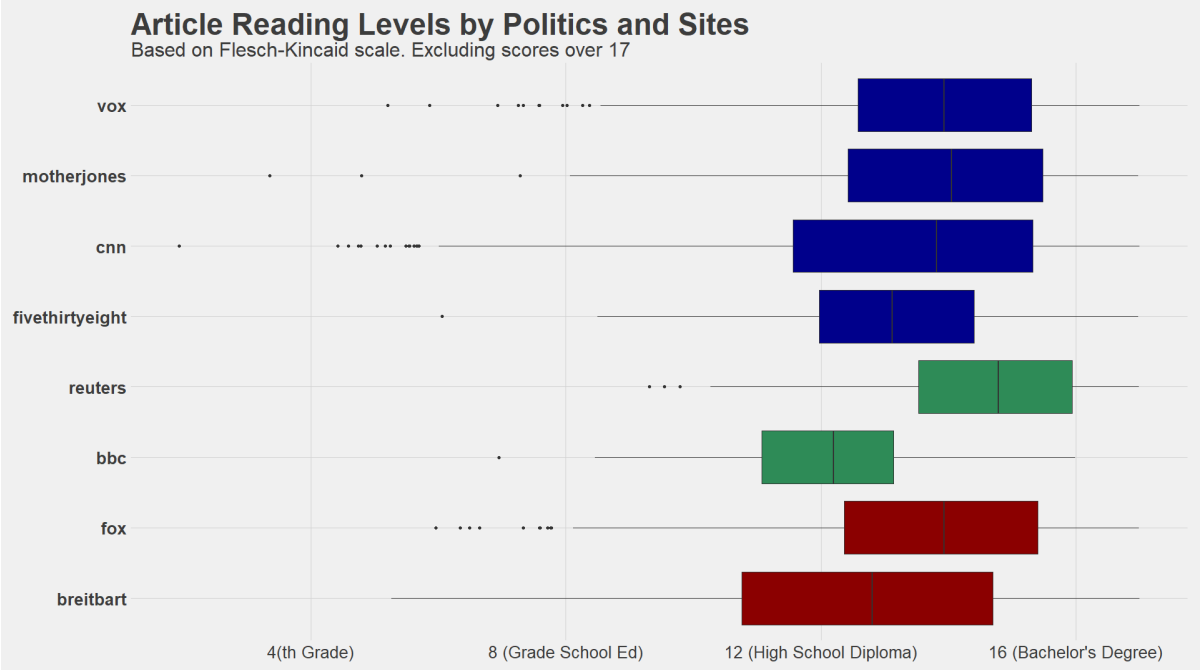


Figure 12: Article reading levels

Figure 13: Word cloud for left leaning articles

Figure 14: Word cloud for right leaning articles

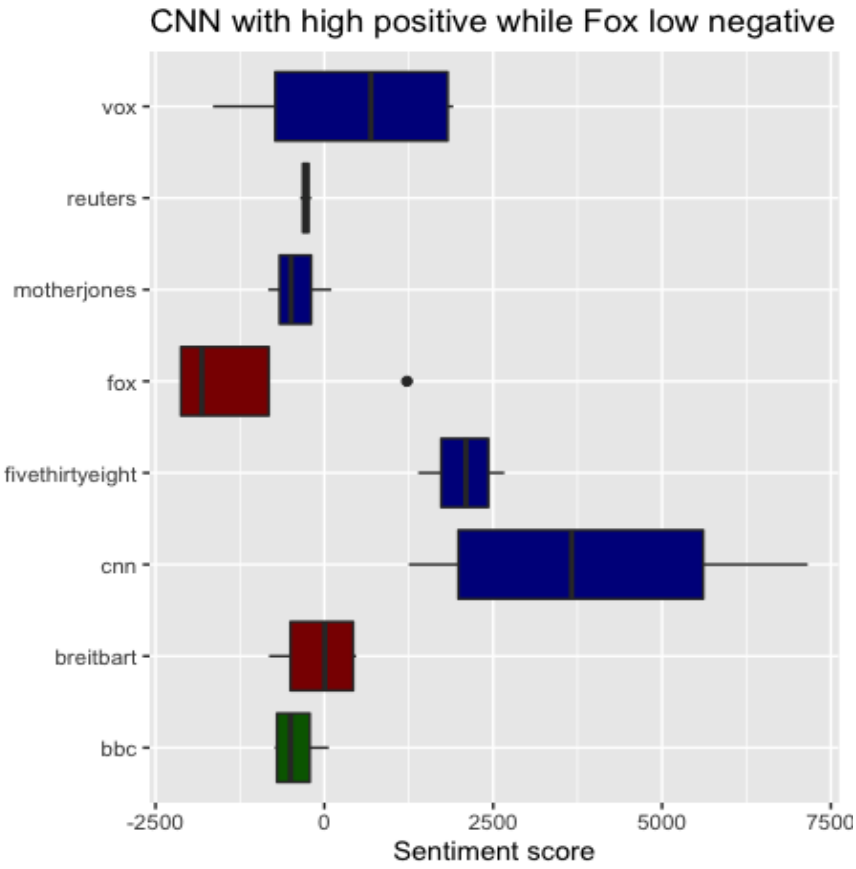


Figure 15: Individual sentiment analysis

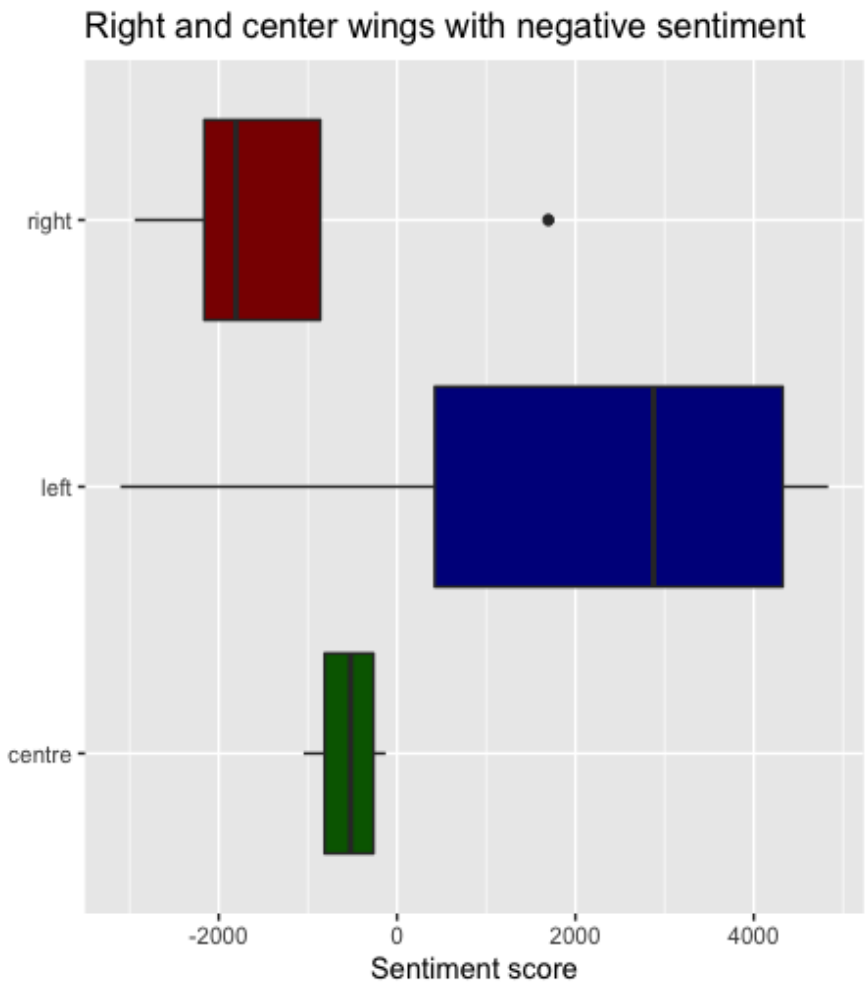


Figure 16: Overall sentiment analysis

Appendix B

Section 1: RNN Code

```
BASE_DIR = ''
GLOVE_DIR = os.path.join(BASE_DIR, 'glove.6B')
TEXT_DATA_DIR = os.path.join(BASE_DIR, '20_newsgroup')
MAX_SEQUENCE_LENGTH = 200
MAX_NUM_WORDS = 20000
EMBEDDING_DIM = 300
VALIDATION_SPLIT = 0.2
# first, build index mapping words in the embeddings set
# to their embedding vector

print('Indexing word vectors.')
```

```
embeddings_index = {}
with open('wiki.en.vec', 'r') as f:
      for line in f:
      word, coefs = line.split(maxsplit=1)
      coefs = np.fromstring(coefs, 'f', sep=' ')
      if len(coefs) < 300:
            continue
      embeddings_index[word] = coefs

print('Found %s word vectors.' % len(embeddings_index))
# second, prepare text samples and their labels
print('Processing text dataset')

texts = []  # list of text samples
labels_index = {'left': 0, 'right': 1, 'centre': 2}  # dictionary mapping
label name to numeric id
labels = []  # list of label ids
train_split = pd.read_csv("train_split.csv")
texts = train_split['article_text']
labels = train_split['political_lean'].map(labels_index)

print('Found %s texts.' % len(texts))

# finally, vectorize the text samples into a 2D integer tensor
tokenizer = Tokenizer(num_words=MAX_NUM_WORDS)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)

word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))


data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)

labels = to_categorical(np.asarray(labels))
print('Shape of data tensor:', data.shape)
print('Shape of label tensor:', labels.shape)

# split the data into a training set and a validation set
indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]
```

```
num_validation_samples = int(VALIDATION_SPLIT * data.shape[0])

x_train = data[:-num_validation_samples]
y_train = labels[:-num_validation_samples]
x_val = data[-num_validation_samples:]
y_val = labels[-num_validation_samples:]


print('Preparing embedding matrix.')

# prepare embedding matrix
num_words = min(MAX_NUM_WORDS, len(word_index) + 1)
embedding_matrix = np.zeros((num_words, EMBEDDING_DIM))
for word, i in word_index.items():
      if i >= MAX_NUM_WORDS:
      continue
      embedding_vector = embeddings_index.get(word)
      if embedding_vector is not None:
      # words not found in embedding index will be all-zeros.
      embedding_matrix[i] = embedding_vector

# load pre-trained word embeddings into an Embedding layer
# note that we set trainable = False so as to keep the embeddings fixed
embedding_layer = Embedding(num_words,
                            EMBEDDING_DIM,

      embeddings_initializer=Constant(embedding_matrix),
                            input_length=MAX_SEQUENCE_LENGTH,
                            trainable=False)


sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedded_sequences = embedding_layer(sequence_input)
x = Conv1D(128, 2, activation='relu')(embedded_sequences)
x = MaxPooling1D(2)(x)
x = LSTM(128, recurrent_dropout=0.2)(embedded_sequences)


x = Dense(128, activation='relu')(x)
x = Dropout(0.2)(x)
preds = Dense(len(labels_index), activation='softmax')(x)

model = Model(sequence_input, preds)
```

```
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['acc'])

model.fit(data, labels,
              batch_size=128,
              epochs=50)
```

Section 2: Code for the least popular candidate running for Presidency in the next election

```
counts <- tidy_articles %>%
  semi_join(names)%>%
  count(news_site, word, sort = TRUE)

least_counts <- tidy_articles %>%
  semi_join(names)%>%
  count(news_site, word, sort = TRUE) %>%
  group_by(news_site)%>%
  summarise(least_popular_candidate = min(n))%>%
  ungroup()

least_popular_candidates <-
counts%>%semi_join(least_counts,by=c("n"="least_popular_candidate"))%>%
  rename(number_of_mentions = n)
least_popular_candidates
```

Here, '*least_popular_candidates*' tibble/data frame stores the last name of the least popular candidate running for Presidency in the next election in each news site along with the site's political lean and count of the last name.