Technical Appendix

Andrea Boskovic and Peter Cho

12/18/2018

Abstract:

In order to categorize an article as real or fake, we joined three datasets containing both real and fake news and used a machine learning algorithm. We trained this data based on the number of words, number of positively categorized words, number of negatively categorized words, maximum negative score, maximum positive score, and AFINN lexicon score of the article which represents a range of negativity and positivity of sentiment. We found the article's AFINN value by decomposing the article into individual words and then averaging the AFINN scores of each word in every article. Using the decision tree learning algorithm, we attempted to find the best threshold for our selected predictors that correlates with fake or real news. However, our decision tree only had one node, meaning that the predictors were not useful in predicting an article's validity.

Introduction:

Modern day media does not always make honest claims or provide truthful sources. The ease of access to the internet has led to the widespread creation and distribution of fake news, leaving readers to question the validity of the news they are reading. To prevent further disinformation, we sought to find a pattern between fake and real news. When we were reading news articles that were considered fake, we noticed a trend of extreme word choices and hypothesized that fake articles would contain more emotionally charged words than real articles. Following this assumption, we sought to create a supervised machine learning model that would classify news as real or fake based on the sentiment of the article text. For the supervised model, we needed to create a training dataset and chose datasets containing articles that distinguished between real or fake news based on the validity of the article's sources and the claims stated. We filtered the article text for the sentiment words and created predictors from the sentiment values we obtained. Using these predictors, we trained a decision tree model to find some relationship between the sentiment predictors and article validity but found none. To further investigate why our model failed, we looked at the relationship between each predictor variable and the response variable and found no significant difference between news categorized as real or fake.

Data:

Before importing in the news datasets, we loaded packages in R to work with text-based data. Our initial cluster of packages provided the classic data tidying and wrangling functions. The following cluster included text specific packages such as text mining (TM) to parse through lists of texts and sort through words.

Three datasets containing fake or real news articles and traits about these articles were taken from Kaggle to create a larger dataset. These datasets included information about the validity of the news and had a variable that either binarized news as real or fake or categorized news as some subcategory of real or fake (i.e. bias, conspiracy, bs). We added a variable that converted these subcategories into either real or fake to allow joining of all three datasets. We selected for this binary variable, the subcategory of the binary variable, the article text, the article id, and the article title from each dataset. To eventually perform sentiment analysis, we used a function to separate individual words from a chunk of text and made each word in the article a new variable.

```
# Loading the dataset
fake <- read_csv("fake.csv") #all fake
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   ord_in_thread = col_integer(),
```

```
##   published = col_datetime(format = ""),
##   crawled = col_datetime(format = ""),
##   domain_rank = col_integer(),
##   spam_score = col_double(),
##   replies_count = col_integer(),
##   participants_count = col_integer(),
##   likes = col_integer(),
##   comments = col_integer(),
##   shares = col_integer()
## )

## See spec(...) for full column specifications.
```

```
real <- read_csv("Articles.csv") #all real
```

```
## Parsed with column specification:
## cols(
##   Article = col_character(),
##   Date = col_character(),
##   Heading = col_character(),
##   NewsType = col_character()
## )
```

```
new_ds <- read_csv("data.csv") #combination of real and fake
```

```
## Parsed with column specification:
## cols(
##   URLs = col_character(),
##   Headline = col_character(),
##   Body = col_character(),
##   Label = col_integer()
## )
```

```
fake_type <- c("fake", "satire", "bias", "bs", "conspiracy", "state", "junksci", "hate")
real_type <- c("sports", "business")

# Merging the datasets and removing unnecessary columns
real <- real %>%
  mutate(binary_type = ifelse(NewsType %in% fake_type, 0, 1)) #now fake = 0 and real = 1
fake <- fake %>%
  mutate(binary_type = ifelse(type %in% fake_type, 0, 1)) #now fake = 0 and real = 1
new_ds <- new_ds %>%
  filter(Label == 1)
real <- full_join(real, new_ds, by = c("Heading" = "Headline", "Article" = "Body", "binary_type" = "Labe
real <- real %>%
  mutate(id = as.character(seq(1:4564))) %>%
  mutate(realtype = "real")

# Making a combined dataset with both fake and real articles and selecting only for the uuid (unique id
combined <- full_join(fake, real, by = c("text" = "Article", "title" = "Heading", "uuid" = "id", "binary
  select(uuid, binary_type, type, title, text)

# Making a tidy dataset where we have the the words in their own column for facilitated data analysis a
tidy_combined <- combined %>%
  unnest_tokens(word, text)
```

We observed the number of fake and real type news and saw that 73% of the dataset consisted of fake news.

```r
# This allows us to see how many observations are in each type of fake news.
combined %>%
  group_by(type) %>%
  summarize(n = n())
```

```
## # A tibble: 9 x 2
##   type           n
##   <chr>      <int>
## 1 bias         443
## 2 bs         11492
## 3 conspiracy   430
## 4 fake          19
## 5 hate         246
## 6 junksci      102
## 7 real        4564
## 8 satire       146
## 9 state        121
```

```r
typetotals <- combined %>%
  group_by(type) %>%
  summarize(n = n())
```

In order to observe any trend in the emotions of news articles, we used sentiment lexicons. These lexicons are datasets containing lists of words with an associated sentiment and/or metric to rate the positivity or negativity of the word. The three lexicons we applied to our dataset were nrc, bing, and AFINN. The nrc lexicon categorized words based on the eight basic emotions (fear, disgust, trust, anger, anticipation, surprise, sadness, joy) and included a binary sentiment that was positive or negative.

```r
# What are the most common words for each basic emotion?
# We will use the nrc lexicon to categorize each documented word into on of the basic human emotions ca

# Anger
nrc_anger <- get_sentiments("nrc") %>%
  filter(sentiment == "anger")

tidy_combined %>%
  inner_join(nrc_anger) %>%
  count(word, sort = TRUE)
```

```
## Joining, by = "word"
```

```
## # A tibble: 1,220 x 2
##    word         n
##    <chr>    <int>
##  1 vote      4969
##  2 money     4835
##  3 force     3189
##  4 court     2721
##  5 attack    2548
##  6 defense   2242
##  7 death     2176
##  8 bad       2175
##  9 politics  2058
## 10 fight     2054
## # ... with 1,210 more rows
```

```r
# Fear
nrc_fear <- get_sentiments("nrc") %>%
  filter(sentiment == "fear")

tidy_combined %>%
  inner_join(nrc_fear) %>%
  count(word, sort = TRUE)
```

```
## Joining, by = "word"

## # A tibble: 1,430 x 2
##    word           n
##    <chr>      <int>
##  1 government 11656
##  2 war         9845
##  3 military    5880
##  4 police      4902
##  5 change      4442
##  6 case        4177
##  7 force       3189
##  8 court       2721
##  9 attack      2548
## 10 problem     2381
## # ... with 1,420 more rows
```

```r
# Anticipation
nrc_anticipation <- get_sentiments("nrc") %>%
  filter(sentiment == "anticipation")

tidy_combined %>%
  inner_join(nrc_anticipation) %>%
  count(word, sort = TRUE)
```

```
## Joining, by = "word"

## # A tibble: 816 x 2
##    word              n
##    <chr>         <int>
##  1 time          14159
##  2 white          6547
##  3 public         6039
##  4 good           5802
##  5 long           5706
##  6 vote           4969
##  7 money          4835
##  8 investigation  3968
##  9 top            3822
## 10 continue       3439
## # ... with 806 more rows
```

```r
# Trust
nrc_trust <- get_sentiments("nrc") %>%
  filter(sentiment == "trust")

tidy_combined %>%
  inner_join(nrc_trust) %>%
```

```r
  count(word, sort = TRUE)
```

```
## Joining, by = "word"
```

```
## # A tibble: 1,191 x 2
##    word          n
##    <chr>     <int>
##  1 president 12344
##  2 united     7803
##  3 white      6547
##  4 good       5802
##  5 law        5181
##  6 system     5088
##  7 vote       4969
##  8 police     4902
##  9 money      4835
## 10 fact       4673
## # ... with 1,181 more rows
```

```r
# Surprise
nrc_surprise <- get_sentiments("nrc") %>%
  filter(sentiment == "surprise")

tidy_combined %>%
  inner_join(nrc_surprise) %>%
  count(word, sort = TRUE)
```

```
## Joining, by = "word"
```

```
## # A tibble: 518 x 2
##    word      n
##    <chr> <int>
##  1 trump 23953
##  2 good   5802
##  3 vote   4969
##  4 money  4835
##  5 deal   2802
##  6 death  2176
##  7 leave  2080
##  8 hope   1902
##  9 young  1859
## 10 shot   1604
## # ... with 508 more rows
```

```r
# Sadness
nrc_sadness <- get_sentiments("nrc") %>%
  filter(sentiment == "sadness")

tidy_combined %>%
  inner_join(nrc_sadness) %>%
  count(word, sort = TRUE)
```

```
## Joining, by = "word"
```

```
## # A tibble: 1,151 x 2
##    word         n
##    <chr>    <int>
```

```
## 1 vote      4969
## 2 black     4196
## 3 case      4177
## 4 problem   2381
## 5 lost      2260
## 6 tax       2211
## 7 death     2176
## 8 bad       2175
## 9 leave     2080
## 10 violence 1955
## # ... with 1,141 more rows
```

```r
# Joy
nrc_joy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")

tidy_combined %>%
  inner_join(nrc_joy) %>%
  count(word, sort = TRUE)
```

```
## Joining, by = "word"
```

```
## # A tibble: 668 x 2
##    word      n
##    <chr> <int>
##  1 white  6547
##  2 good   5802
##  3 vote   4969
##  4 money  4835
##  5 found  4192
##  6 share  3090
##  7 deal   2802
##  8 food   2756
##  9 pay    2339
## 10 true   2234
## # ... with 658 more rows
```

```r
# Disgust
nrc_disgust <- get_sentiments("nrc") %>%
  filter(sentiment == "disgust")

tidy_combined %>%
  inner_join(nrc_disgust) %>%
  count(word, sort = TRUE)
```

```
## Joining, by = "word"
```

```
## # A tibble: 1,023 x 2
##    word          n
##    <chr>     <int>
##  1 john       3108
##  2 congress   2473
##  3 death      2176
##  4 bad        2175
##  5 criminal   1805
##  6 illegal    1756
##  7 powerful   1611
```

```
##  8 corruption  1571
##  9 finally     1442
## 10 remains     1244
## # ... with 1,013 more rows
```

The bing lexicon solely applied a binary sentiment of either positive or negative. With this lexicon, we attempted to find differences in the total sentiment value between the subcategory of news. The total sentiment value was calculated by subtracting the total number of positive words by the number of negative words in each category.

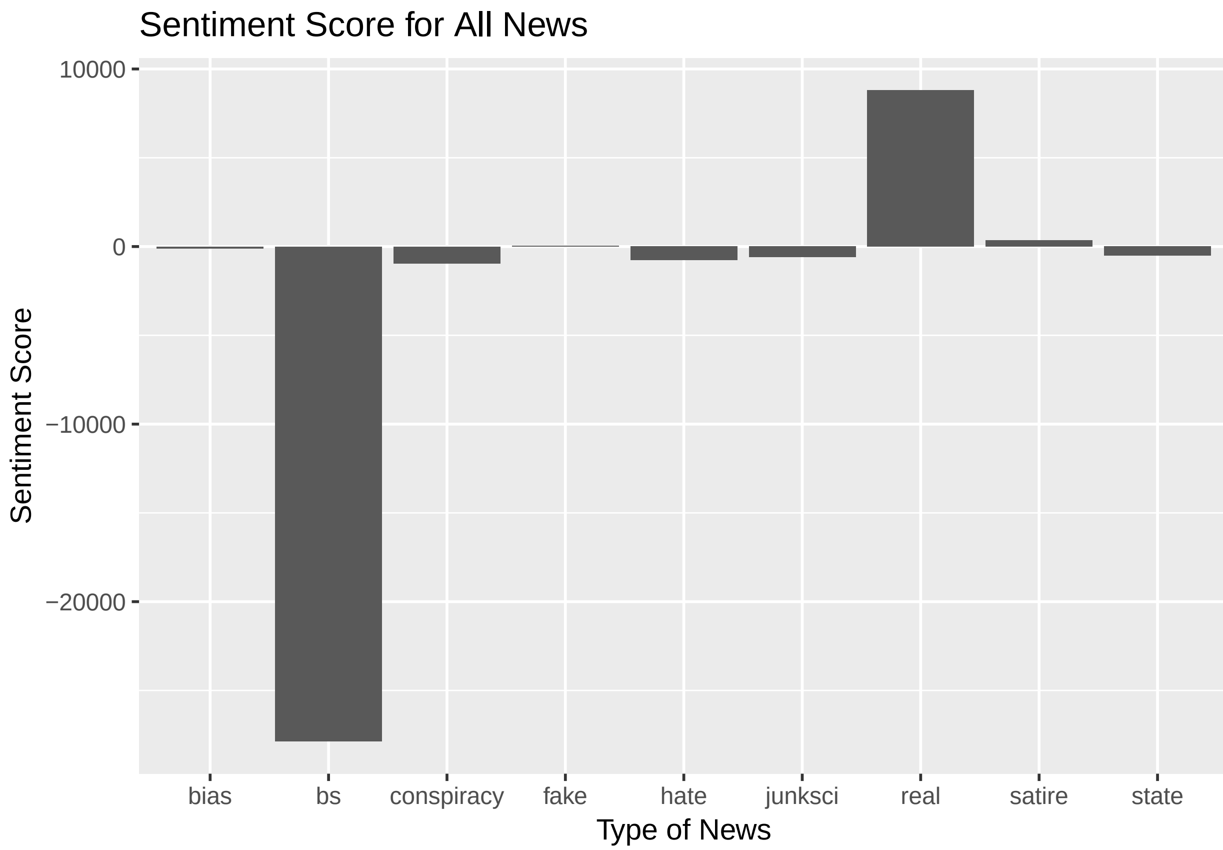
```r
# Find net sentiment for each type of fake news documented in the dataset using the bing lexicon. The b
# Note that some types, such as bs (> 400,000), have more corresponding observations than other types, 
combined_sentiment <- tidy_combined %>%
  inner_join(get_sentiments("bing")) %>%
  count(type, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
```

```
## Joining, by = "word"
```

```r
combined_sentiment
```

```
## # A tibble: 9 x 4
##   type       negative positive sentiment
##   <chr>         <dbl>    <dbl>     <dbl>
## 1 bias           5422     5322      -100
## 2 bs           247391   219536    -27855
## 3 conspiracy     4805     3851      -954
## 4 fake            148      199        51
## 5 hate           8765     7998      -767
## 6 junksci        3070     2469      -601
## 7 real          45896    54690      8794
## 8 satire         1148     1487       339
## 9 state          1215      704      -511
```

```r
# Plot of the sentiment score for each type of news
ggplot(combined_sentiment, aes(x = type, y = sentiment)) + geom_col() + labs(title = "Sentiment Score fo
```



Based on this visualization of the data, we observed a markedly negative value for the "bs" subcategory of fake news and a clear positive value for real news.

```
# We can also get the sentiment score on a scale of -5 to 5 from the AFINN lexicon. The AFINN lexicon h
afinn <- tidy_combined %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(type) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(method = "AFINN")
```

```
## Joining, by = "word"
```

```
afinn
```

```
## # A tibble: 9 x 3
##   type       sentiment method
##   <chr>          <int> <chr>
## 1 bias           -1507 AFINN
## 2 bs            -62021 AFINN
## 3 conspiracy     -1846 AFINN
## 4 fake             108 AFINN
## 5 hate           -1625 AFINN
## 6 junksci           41 AFINN
## 7 real           28457 AFINN
## 8 satire           868 AFINN
## 9 state          -1089 AFINN
```

Since the nrc and bing lexicons could binarize a word as either positive or negative, we measured the ratio of positive to negative words in each lexicon to test for any skew. Both lexicons have more negative words than positive words, but the bing lexicon has a higher ratio of negative to positive words than the nrc lexon.

```
# Positive and negative words in nrc lexicon
get_sentiments("nrc") %>%
     filter(sentiment %in% c("positive",
                              "negative")) %>%
  count(sentiment)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   3324
## 2 positive   2312
```

```
# Positive and negative words in bing lexicon
get_sentiments("bing") %>%
  count(sentiment)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   4782
## 2 positive   2006
```

Another factor we observed in our data was the frequency of words. We removed words that were frequent in most text and provided little information by filtering out stop words (i.e. "the", "I", "a", etc.). We then selected for words that were part of the bing lexicon and created a word cloud to visualize the frequency and divide between positive and negative words. We then graphed a bar chart to record the number of times a word was mentioned and found the frequency of words "trump" and "like" to be most common among positive words. We assumed that the high frequency was in reference to President Trump and the use of "like" to not mean the things one prefers.

```
# Counting the most frequently appearing words and which sentiment they correspond to (positive or nega
bing_word_counts <- tidy_combined %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

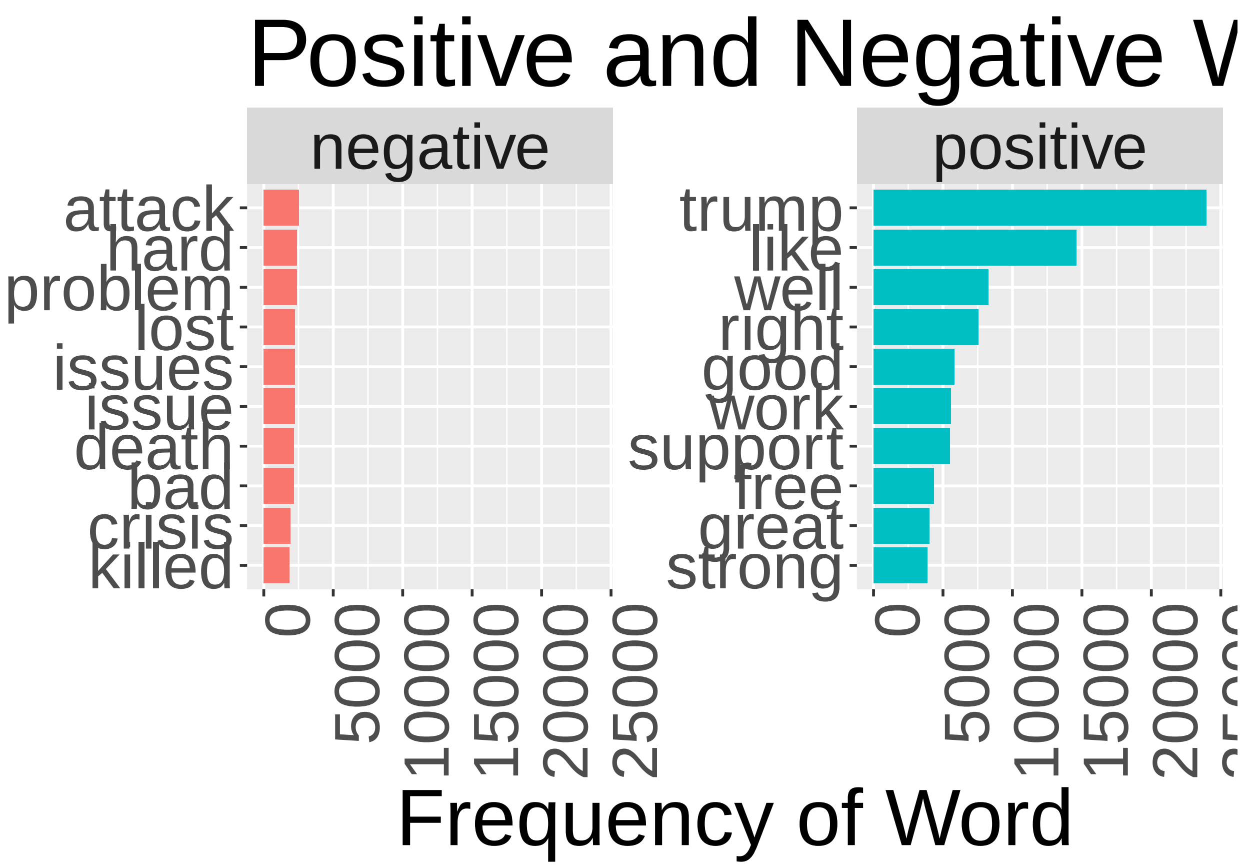
```
## Joining, by = "word"
```

```
bing_word_counts
```

```
## # A tibble: 5,552 x 3
##    word    sentiment     n
##    <chr>   <chr>     <int>
##  1 trump   positive  23953
##  2 like    positive  14612
##  3 well    positive   8250
##  4 right   positive   7530
##  5 good    positive   5802
##  6 work    positive   5544
##  7 support positive   5504
##  8 free    positive   4327
##  9 great   positive   4007
## 10 strong  positive   3862
## # ... with 5,542 more rows
```

```r
bing_word_counts %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  ggtitle("Positive and Negative Word Frequency") +
  labs(y = "Frequency of Word",
       x = NULL) +
  theme(text = element_text(size=30),
        axis.text.x = element_text(angle=90, hjust=1)) +
  coord_flip()
```

```
## Selecting by n
```

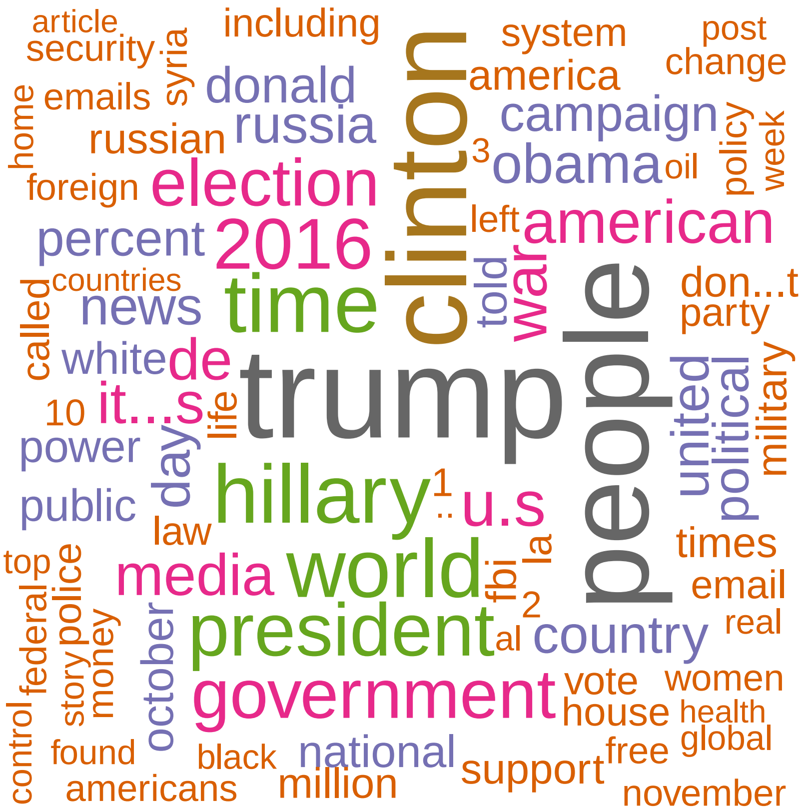



```r
# Wordcloud with most frqeuently appearing words
tidy_combined %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(words = word, freq = n, max.words = 100, min.freq = 1, random.order=FALSE, rot.per = 0
```

```
## Joining, by = "word"

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'it's' in 'mbcsToSbcs': dot substituted for <e2>
```

```
## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'it’s' in 'mbcsToSbcs': dot substituted for <80>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'it’s' in 'mbcsToSbcs': dot substituted for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'it’s' in 'mbcsToSbcs': dot substituted
## for <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'it’s' in 'mbcsToSbcs': dot substituted
## for <80>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'it’s' in 'mbcsToSbcs': dot substituted
## for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2019

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'don’t' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'don’t' in 'mbcsToSbcs': dot substituted for <80>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'don’t' in 'mbcsToSbcs': dot substituted for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted
## for <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted
## for <80>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted
## for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2019

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : presidential could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : information could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : washington could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : international could not be fit on page. It will not be plotted.

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## ' ' in 'mbcsToSbcs': dot substituted for <d0>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## ' ' in 'mbcsToSbcs': dot substituted for <b2>
```

```
## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on ' ' in 'mbcsToSbcs': dot substituted for
## <d0>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on ' ' in 'mbcsToSbcs': dot substituted for
## <b2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+0432

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : report could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : investigation could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : democratic could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : york could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : days could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : economic could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : strong could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : china could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : human could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : history could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : anti could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : team could not be fit on page. It will not be plotted.
```




```r
# Wordcloud faceted into positive and negative with color (blue corresponds to a negative sentiment whi
tidy_combined %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("blue", "orange"),
                   max.words = 100)
```

```
## Joining, by = "word"
```

Results:

```r
# Now, it is time to start the machine learning aspect of the project.
# Using the AFINN lexicon to append the sentiment score of each word to a new dataset called tidy_combi
tidy_combined_a <- tidy_combined %>%
  inner_join(get_sentiments("afinn"))
```

```
## Joining, by = "word"
```

In order to predict whether an article was real or fake, we decided to use a decision tree to categorize articles as real or fake based on various predictors. Our first model only included average score, which is an average composite score of all the words in the article based on the AFINN score of each words, to predict type of news. In other words, the average score of a word represents the strength of the words negativity or positivity on a scale of negative five to five, and the average score for the article is the sum of the scores for each word that has a score associated to it in the AFINN lexicon divided by the amount of words present in the article with an associate AFINN lexicon score. We then decided to refine our model by adding more predictors. Our next model used the number of negative words and number of positive words in an article, along with the article's average score, to predict whether it would be real or fake. This model, like our first model, resulted in a decision tree with one node. Adding the number of words present in the article, similarly, resulted in a decision tree with one node. Since we saw a seemingly strong relationship between sentiment score from the AFINN lexicon and the type of news (i.e., real, bs, conspiracy) in our data visualization, we added a two more predictors that were indicative of sentiment: negative score and positive score. These values represent the total negativity and positivity, respectively, of an article. We calculated this by filtering for all negatively and positively scored words, according to the AFINN lexicon, in each article, and then summing the scores of all the negative words in an article and all the positive words in an article.

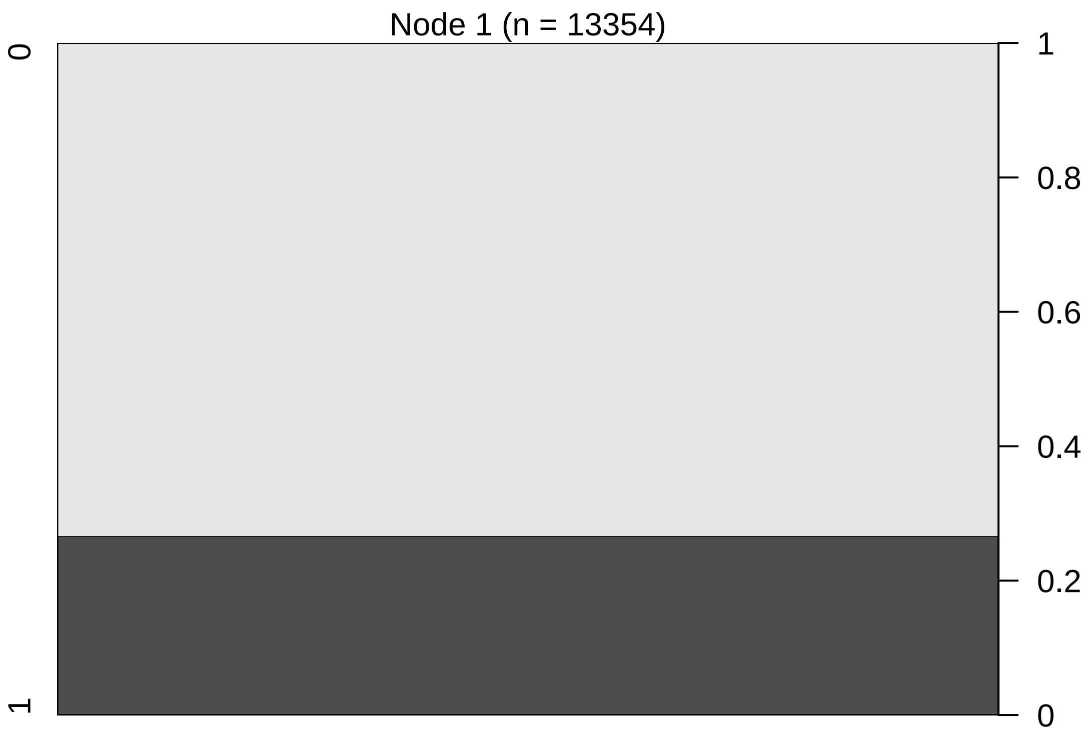
```r
# Categorize article as positive or negative overall based on the average of the AFINN score of the wor
tidy_combined_final <- tidy_combined_a %>%
  select(uuid, score, binary_type) %>%
  group_by(uuid) %>%
  summarise(n_words = n(), avgscore = sum(score) / n_words,
            type = mean(binary_type),
            positive_score = sum(score[score > 0]),
            negative_score = sum(score[score < 0]),
            n_positive = sum(score > 0),
            n_negative = sum(score < 0)
            ) %>%
  mutate(articlesent = ifelse(avgscore < 0, "Negative", "Positive")) %>%
  mutate(txt_type = as.factor(type)) %>%
  select(-type)
tidy_combined_final
```

```
## # A tibble: 16,693 x 9
##    uuid  n_words avgscore positive_score negative_score n_positive
##    <chr>   <int>    <dbl>          <int>          <int>      <int>
##  1 0005~      21    0.286             19            -13         13
##  2 0020~      24   -0.667             12            -28          7
##  3 0021~      87    0.379            109            -76         49
##  4 002d~      88    0.261             99            -76         50
##  5 0033~       9    0                  8             -8          5
##  6 0033~      58   -0.759             36            -80         20
##  7 0037~      14    0.714             16             -6          8
##  8 0038~      30   -0.667             20            -40          9
##  9 003d~      10    0.7               14             -7          7
## 10 0048~      58    0.172             50            -40         34
## # ... with 16,683 more rows, and 3 more variables: n_negative <int>,
## #   articlesent <chr>, txt_type <fct>
```

```r
tidy_combined_final %>%
  filter(txt_type == 0) %>%
  summarise(n_negative = n())
```

```
## # A tibble: 1 x 1
##   n_negative
##        <int>
## 1      12248
```

```r
# Decision tree training process
n <- nrow(tidy_combined_final)
train_id <- sample(1:n, size = round(n * 0.8))
train <- tidy_combined_final[train_id,]
test <- tidy_combined_final[-train_id,]


tree <- rpart(txt_type ~ avgscore + n_words + n_positive + n_negative + negative_score + positive_score
plot(as.party(tree))
```




```r
tree
```

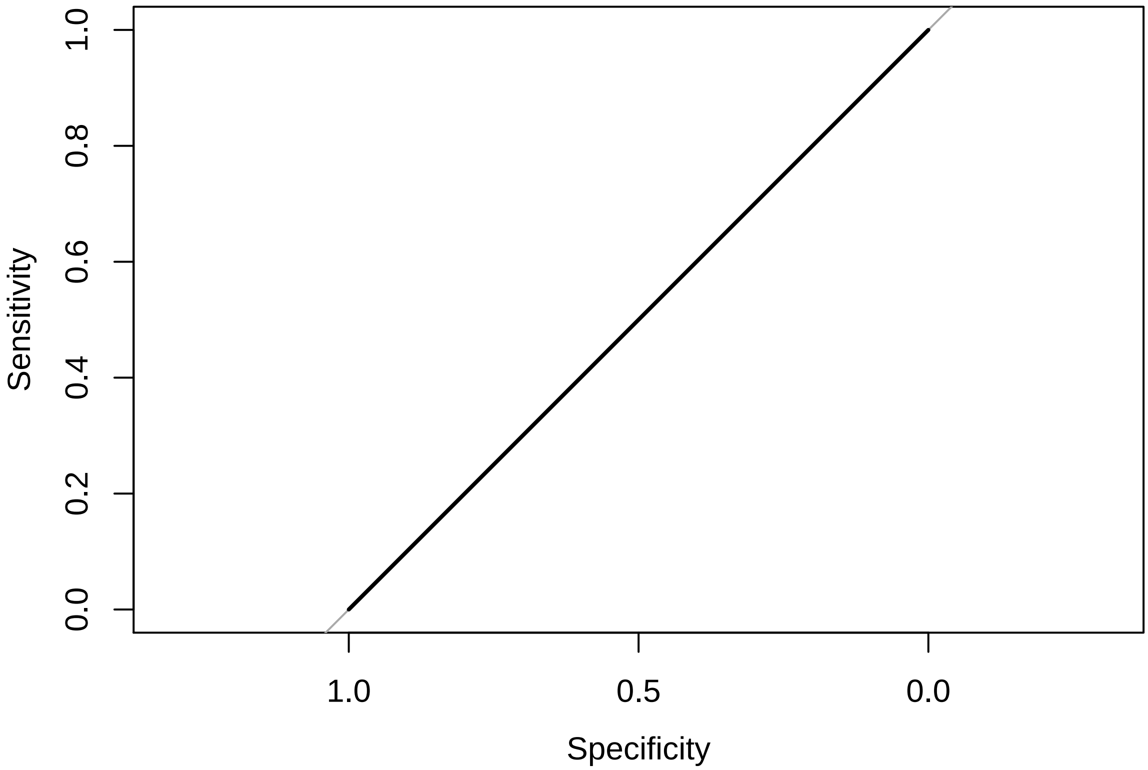
```
## n= 13354
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 13354 3565 0 (0.7330388 0.2669612) *
```

```r
saveRDS(tree, file = "tree.rds")
saveRDS(train, file = "train.rds")
prediction <- predict(tree, test)
```

```r
test <- test %>%
  mutate(prediction = prediction[1])
roc_obj <- roc(test$txt_type, test$prediction)
auc(roc_obj)
```

```
## Area under the curve: 0.5
```

```r
plot(roc_obj)
```



Despite our efforts to make a useful model that can differentiate between real and fake articles, even our final model that used average score, number of words, number of positive words, number of negative words, positive score, and negative score to predict the type of article, our decision tree still had only one node. Likewise, the Sensitivity-Specificity Curve had a straight line and thus an AUC, or area under the curve, of 0.5, the lowest possible AUC value. Our sentiment analysis of articles was unable to predict whether an article was real or fake, so we ultimately found no evidence that our predictors are associated with an article being real or fake.

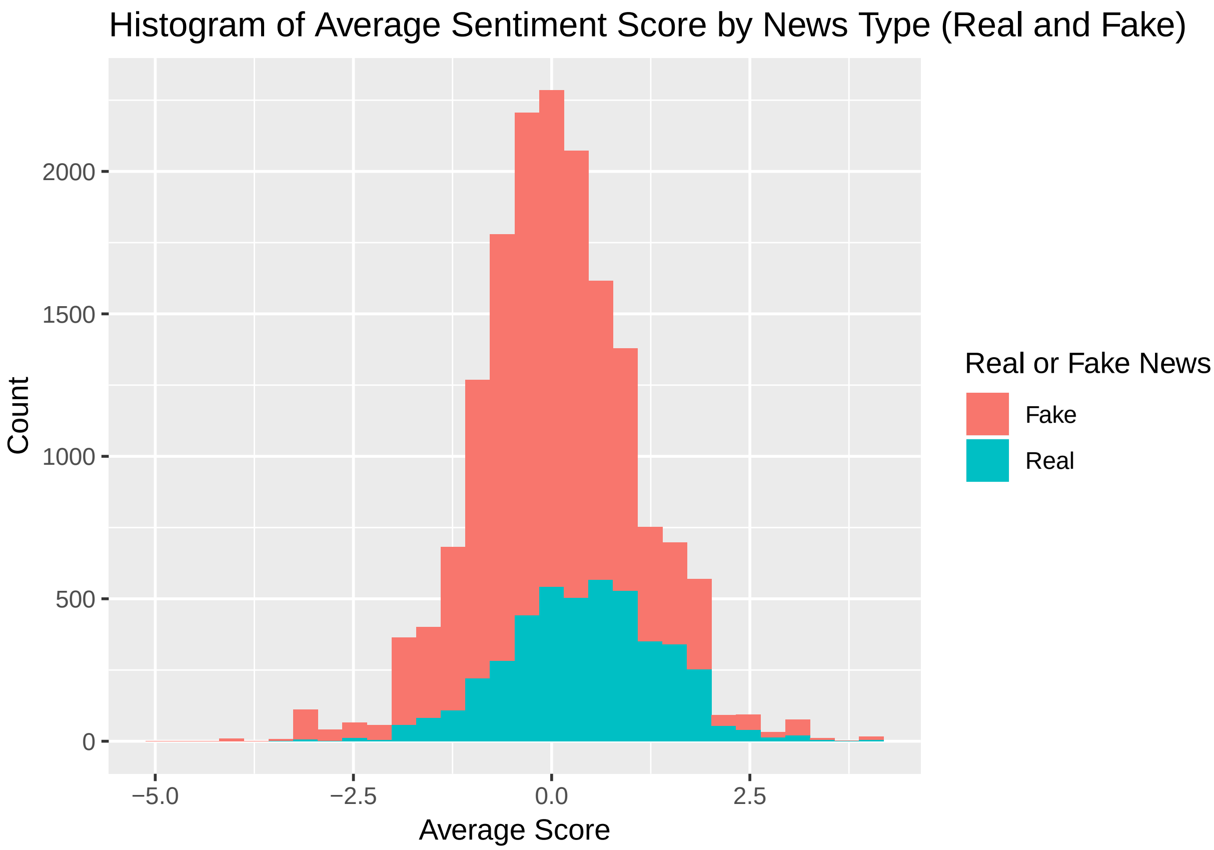
Diagnostics:

The decision tree from our final model had only one node, meaning that our predictors based on sentiment analysis were not useful in predicting whether an article was real or fake. In order to qualify our results, we investigated the relationship between each of our predictor variables with our outcome variable, the validity of the article.

```r
# Why is our model unsuccessful? Below, we will do some exploration using visualizations to display the

# Histogram of Average Sentiment Score by News Type (Real and Fake)
ggplot(tidy_combined_final, aes(x = avgscore, fill = txt_type)) +
  geom_histogram() +
  xlab("Average Score") +
  ylab("Count") +
```

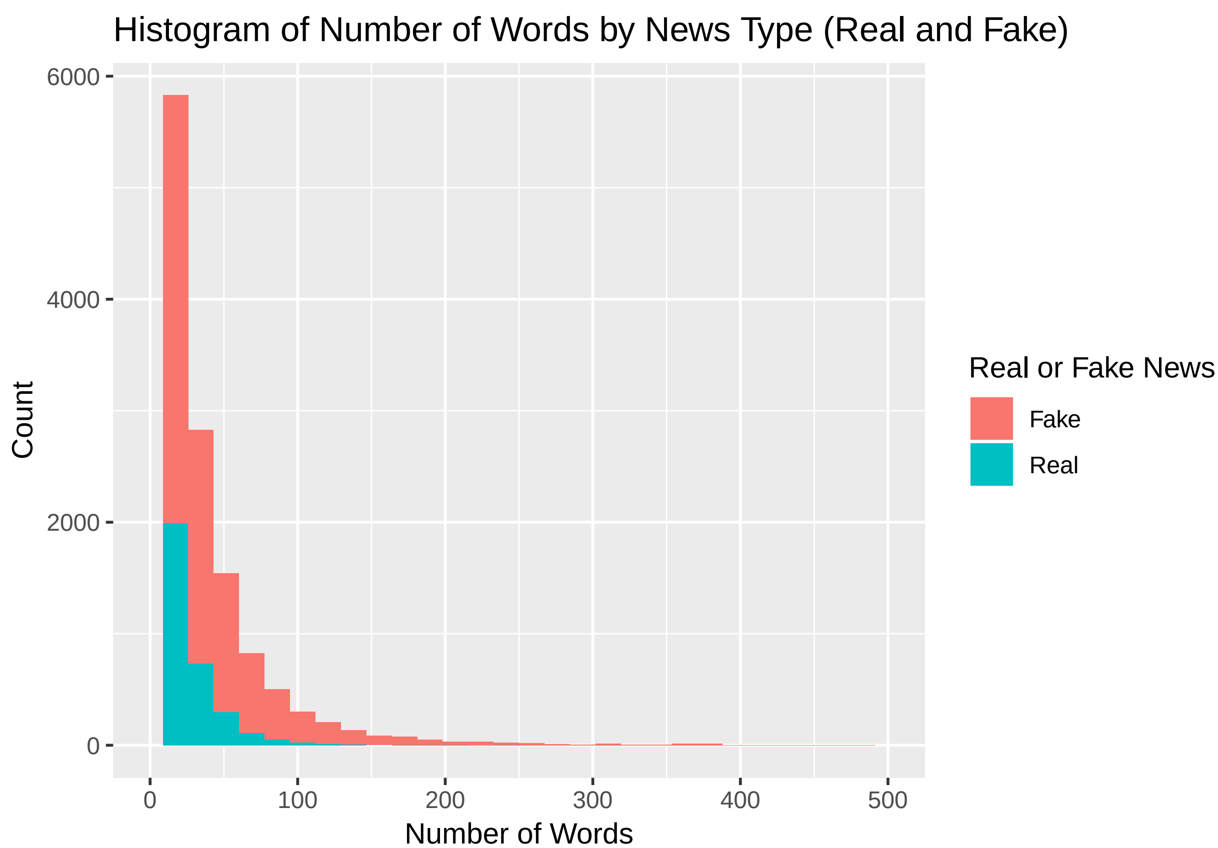
```
  ggtitle("Histogram of Average Sentiment Score by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Histogram of Number of Words by News Type (Real and Fake)
ggplot(tidy_combined_final, aes(x = n_words, fill = txt_type)) +
  geom_histogram() +
  xlim(0, 500) +
  xlab("Number of Words") +
  ylab("Count") +
  ggtitle("Histogram of Number of Words by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 25 rows containing non-finite values (stat_bin).

## Warning: Removed 4 rows containing missing values (geom_bar).
```

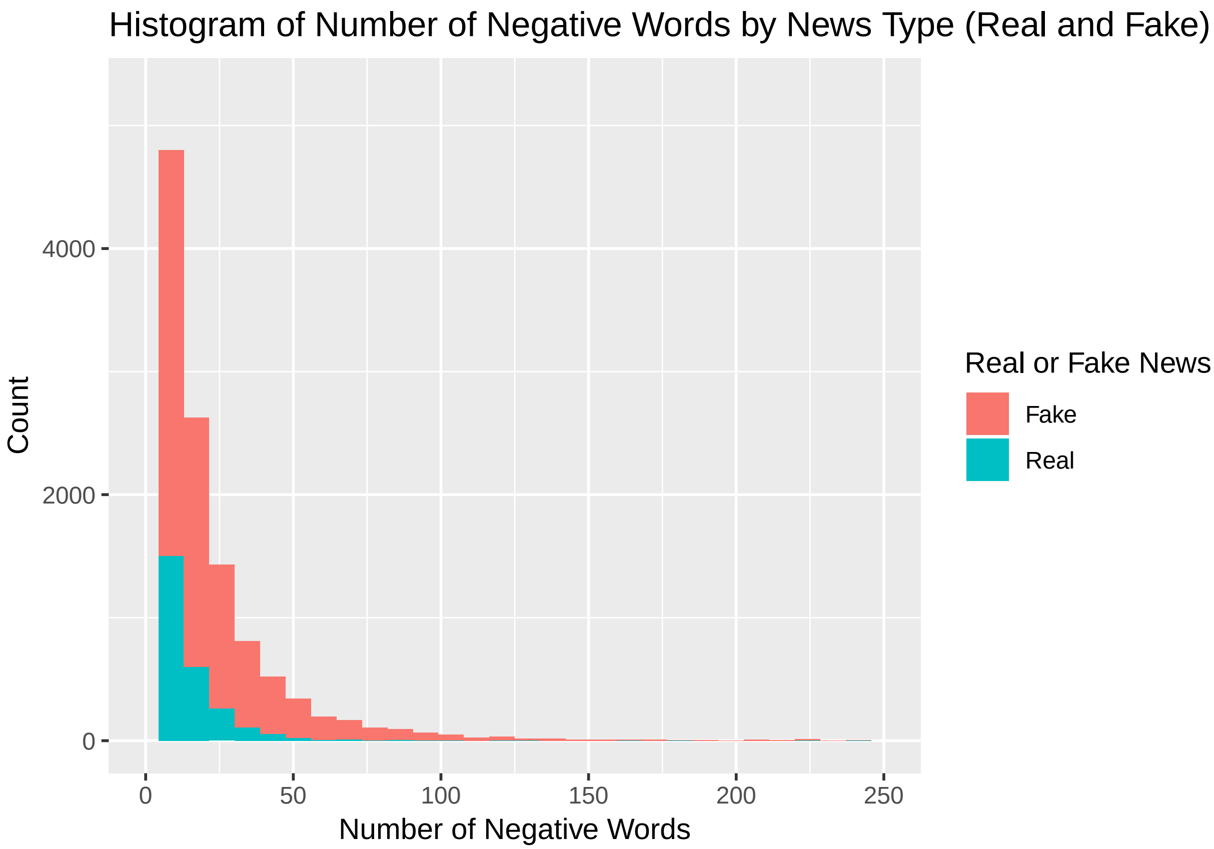



```r
# Histogram of Number of Negative Words by News Type (Real and Fake)
ggplot(tidy_combined_final, aes(x = n_negative, fill = txt_type)) +
  geom_histogram() +
  xlim(0, 250) +
  xlab("Number of Negative Words") +
  ylab("Count") +
  ggtitle("Histogram of Number of Negative Words by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

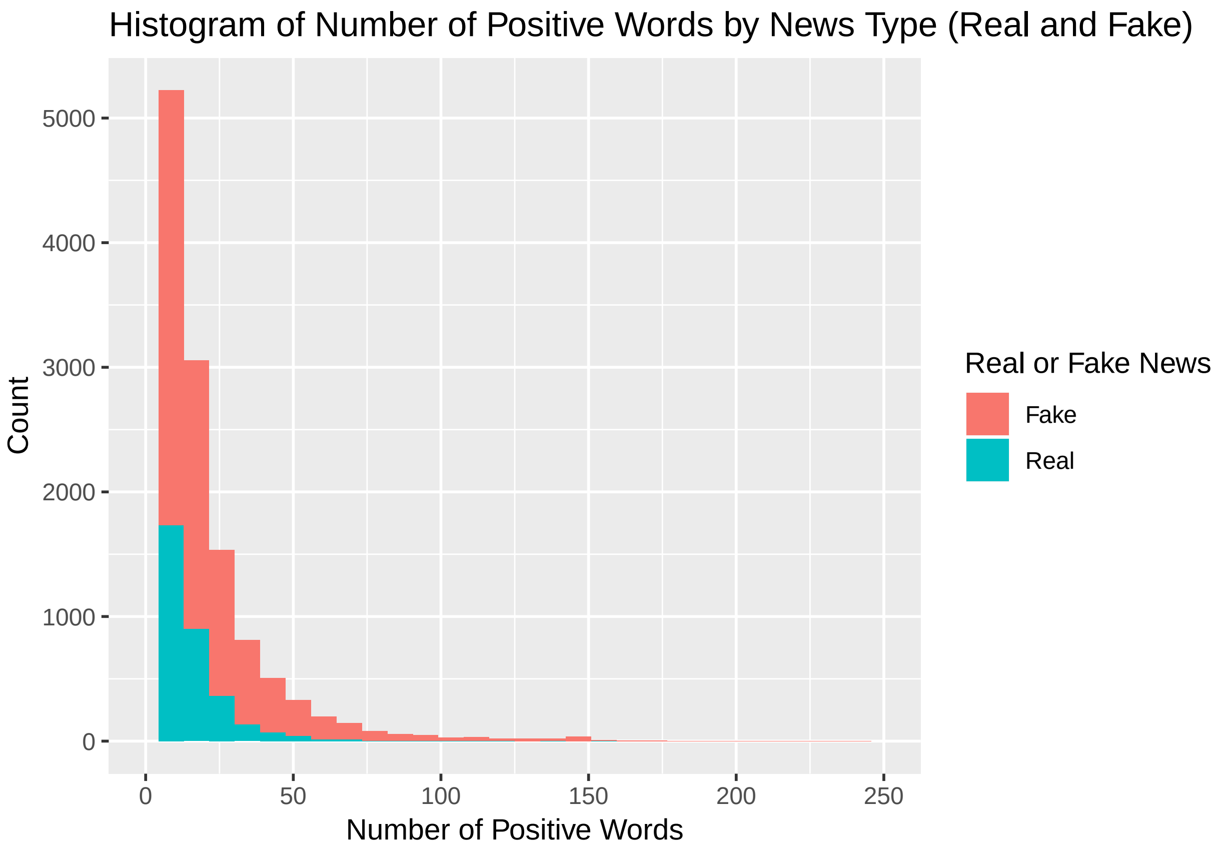
```
## Warning: Removed 27 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 4 rows containing missing values (geom_bar).
```




```
# Histogram of Number of Positive Words by News Type (Real and Fake
ggplot(tidy_combined_final, aes(x = n_positive, fill = txt_type)) +
  geom_histogram() +
  xlim(0, 250) +
  xlab("Number of Positive Words") +
  ylab("Count") +
  ggtitle("Histogram of Number of Positive Words by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 27 rows containing non-finite values (stat_bin).

## Warning: Removed 4 rows containing missing values (geom_bar).
```

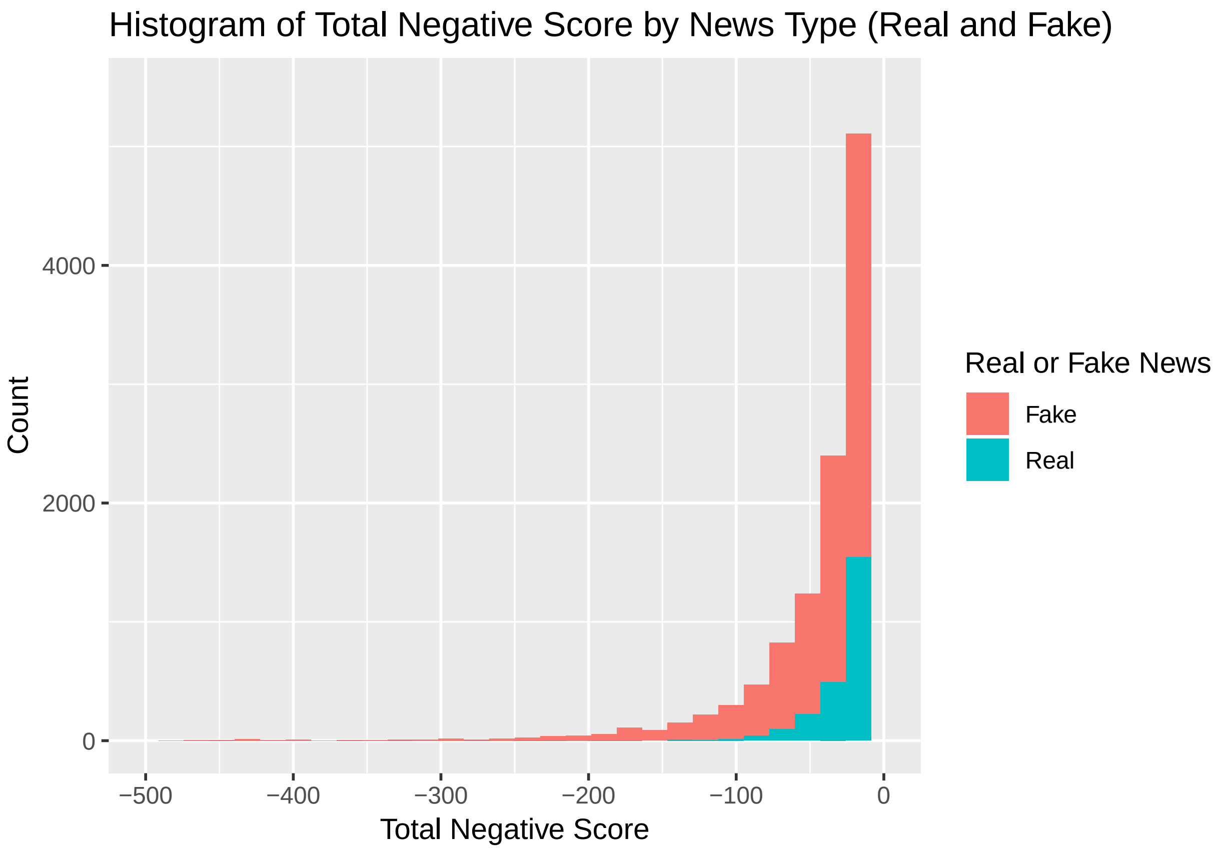



```r
# Histogram of Total Negative Score by News Type (Real and Fake)
ggplot(tidy_combined_final, aes(x = negative_score, fill = txt_type)) +
  geom_histogram() +
  xlim(-500, 0) +
  xlab("Total Negative Score") +
  ylab("Count") +
  ggtitle("Histogram of Total Negative Score by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 25 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 4 rows containing missing values (geom_bar).
```

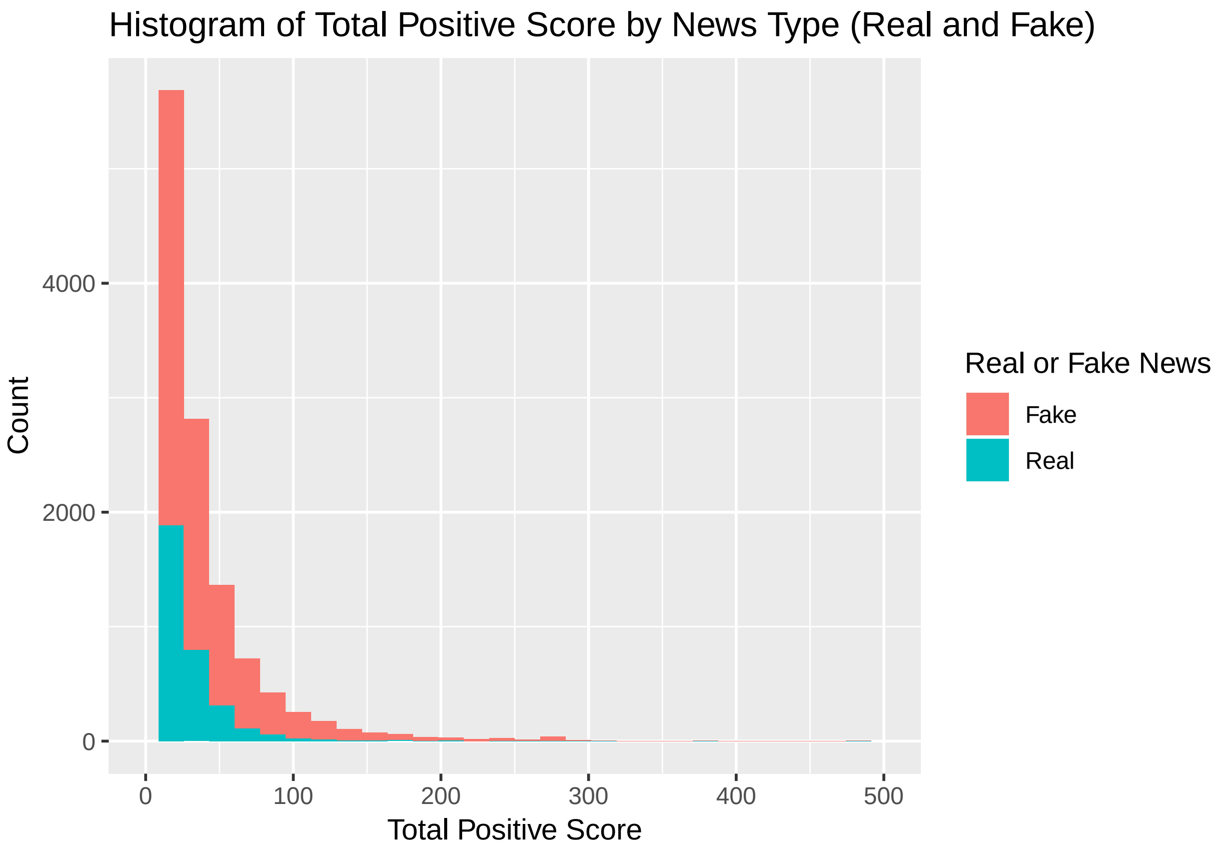



```r
# Histogram of Total Positive Score by News Type (Real and Fake)
ggplot(tidy_combined_final, aes(x = positive_score, fill = txt_type)) +
  geom_histogram() +
  xlim(0, 500) +
  xlab("Total Positive Score") +
  ylab("Count") +
  ggtitle("Histogram of Total Positive Score by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 21 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 4 rows containing missing values (geom_bar).
```



Based on these histograms, we can see that for each predictor, the distributions are similar for both real and fake news. In the histogram of average sentiment score by news type, for example, we see that the distribution of average sentiment for fake news is nearly identical in shape to that of real news. The difference in size, or the area between the peaks of the distributions for real and fake news is due to the higher number of observations in our dataset categorized as fake news. This is feasible because we learned from our exploratory analysis that approximately seventy-three percent of our data is categorized as fake news. Likewise, in the histogram of total positive score by news type shows similarly shaped distributions for real and fake news, with fake news having a larger amount of total observations. Unlike the histogram for average sentiment score, however, this histogram is skewed to the right because most articles, regardless of article validity, seem to have a total positive score of approximately twenty with a decreasing amount of positive scores present in each category of news, fake and real. Similar trends are present in the relationship between the other predictors and news categorized as real or fake.

Conlusion:

In this project, we attempted to classify articles as real or fake based on various indicators of an article's sentiment. Our initial model, which only used the average sentiment score of an article, failed to predict whether it was real or fake. While our final model included more predictors, such as number of words, number of positive words, number of negative words, positive score, and negative score, this too failed at predicting the validity of an article. Both of these models, as well as the intermediate models, led to a decision tree with only one node and thus an area under the curve of the Sensitivity-Specificity curve of only 0.5. Although our use of article sentiment to classify an article as real or fake was not successful, this does not mean it is impossible to classify articles as real or fake using sentiment analysis. Five out of the six predictors in our final model were related to article sentiment, but there are probably ways to categorize sentiment of an article with other predictors. For instance, an interesting technique for future modeling would be to use a different lexicon for sentiment analysis. Despite the fact that we conducted some initial data exploration using the nrc and bing lexicons for sentiment analysis, our final model was entirely based on the AFINN lexicon. There are also more lexicons available for sentiment analysis that could allow us to effectively categorize articles as fake or real. After exploring some literature on methods to categorize the truth of a news article based on the text, we found that someone was able to achieve ninety-five percent accuracy by manually categorizing many articles as fake if the article contained anything not written in a purely factual way. Another paper used grammar, absurdity, and punctuation to predict whether an article was real or fake and got a model with ninety percent accuracy. Since this is not heavily based on sentiment, another possible technique for future analysis and modeling would be to use other factors such as title capitalization and languages used in the article to classify an article as real or fake.