Text Analysis: Twitter News Sources

Nicole Eberle

Contents

Twitter Text Analysis: Compare and Constast Various News Sources

With so many news sources out there, it is common to get push notifications on your phone or see social media posts about the same story from multiple different outlets.Based on this, I wanted to look into how different news accounts were both similar and different by performing text analysis on their most recent tweets.

News sources can be broken down into two main categories - Hard News and Soft News. Hard news includes politics and business while soft news is focused on lifestyle and entertainment, such as sports and celebrity coverage. For this analysis, I will be looking at tweets from both types of sources to see if they have any overlap. The sources used are:
1. enews
2. BuzzFeedNews
3. wsj
4. nytimes
5. Betches_Sup
6. cnn

The main questions I will be looking at are:
1. How similar are twitter accounts of various news outlets?
2. For news outlets that report on similar topics, is the sentiment also the same?

EDA

To start the analysis, I need to load the necessary libraries and import the data. Since I am only concerned with the tweet and the account it came from, I also will pull those columns out of the initial pull and save them to a dataframe.

```
# import libraries
library(tidyverse)
library(tidytext)
library(rtweet)
library(dplyr)
library(topicmodels)
library(ldatuning)
library(textdata)
library(lubridate)

# import data from twitter
# twitterpull <- get_timeline(c("enews", "BuzzFeedNews", "TMZ", "wsj", "nytimes", "Betches_Sup", "cnn")


# export data to file to be able to reproduce same analysis
# saveRDS(twitterpull,"twitterpull.rds")

# or import from downloaded dataset to reproduce project at a later date
 twitterpull <- readRDS("~/DU Spring/Complex Analytics/twitterpull.rds")

# create dataframe with user and tweet
tweets <- as.data.frame(twitterpull %>%
  select(screen_name,text))

# rename screen_name to be Account
colnames(tweets)<- c("Account","text")
```

Before I analyze the data, it is important to make sure I know what the data set contains.

First, Let's look at how many tweets I pulled from each twitter account.

```
tweets %>%
  count(Account)
```

```
       Account    n
1  Betches_Sup 1685
2 BuzzFeedNews 2652
3          CNN 3178
4        enews 3160
5      nytimes 2713
6          TMZ 3164
7          WSJ 3178
```

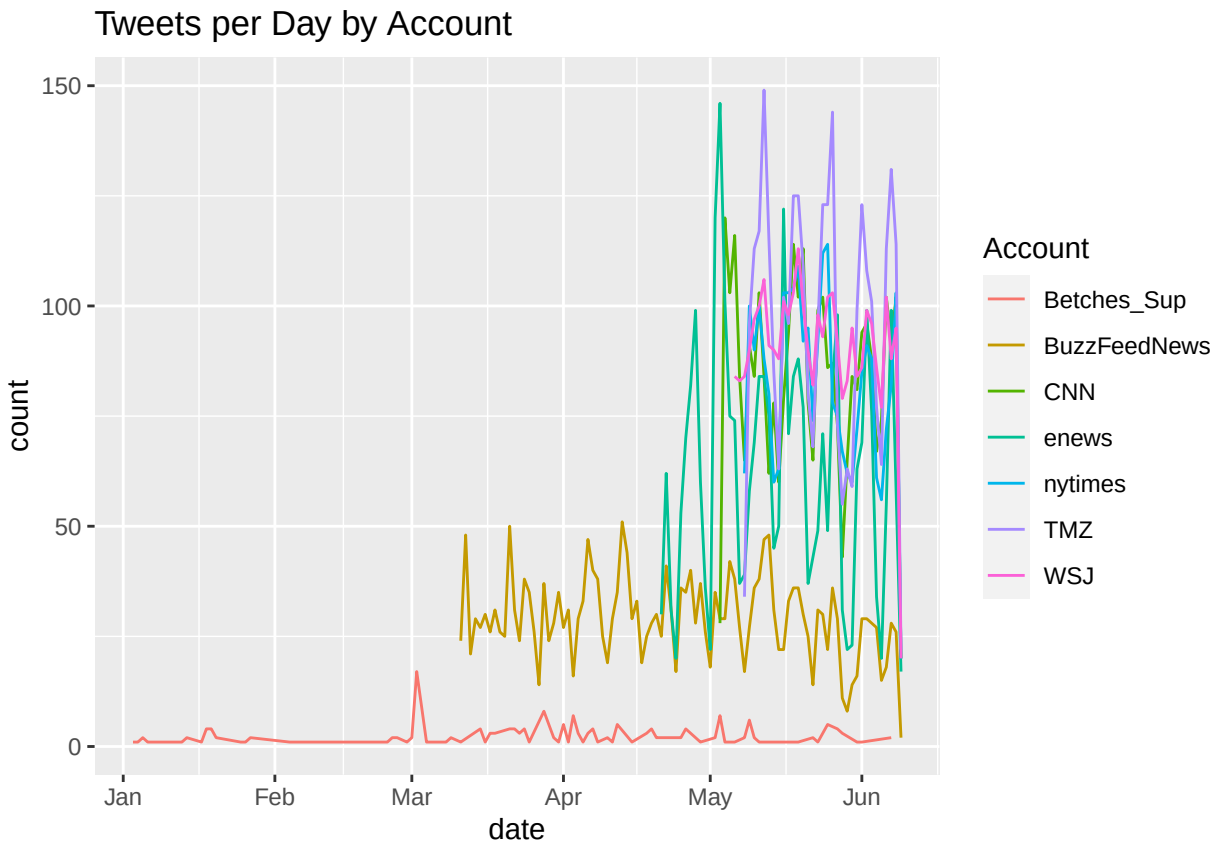
I can see here that Betches_Sup has significantly less tweets in the data set than the other sources. This probably shows us something about how frequently they are tweeting compared to the other accounts and also is likely impacted by the fact that I excluded retweets since much of their content is reacting to news stories. This shows a difference right off the bat in how Betches Sup interacts with and engages their followers.

I can continue to look at the tweet frequency of each account by created a time series plot. To do this, first I will need to make a dataframe that includes information about when each tweet was sent.

```
# create data frame with date information
tweetsdate <- as.data.frame(twitterpull %>%
  select(screen_name,text,created_at))

# rename Account to be account
colnames(tweetsdate)<- c("Account","text", "created_at")

# extract information from created at column
tweetsdate$date <- date(tweetsdate$created_at)
tweetsdate$hour <- hour(tweetsdate$created_at)
tweetsdate$year <- year(tweetsdate$created_at)
```

Now that I have set up the dataframe, I are able to create graphs that show different characteristics about the tweeting habits of each account. First, I will look at tweets per day by account.

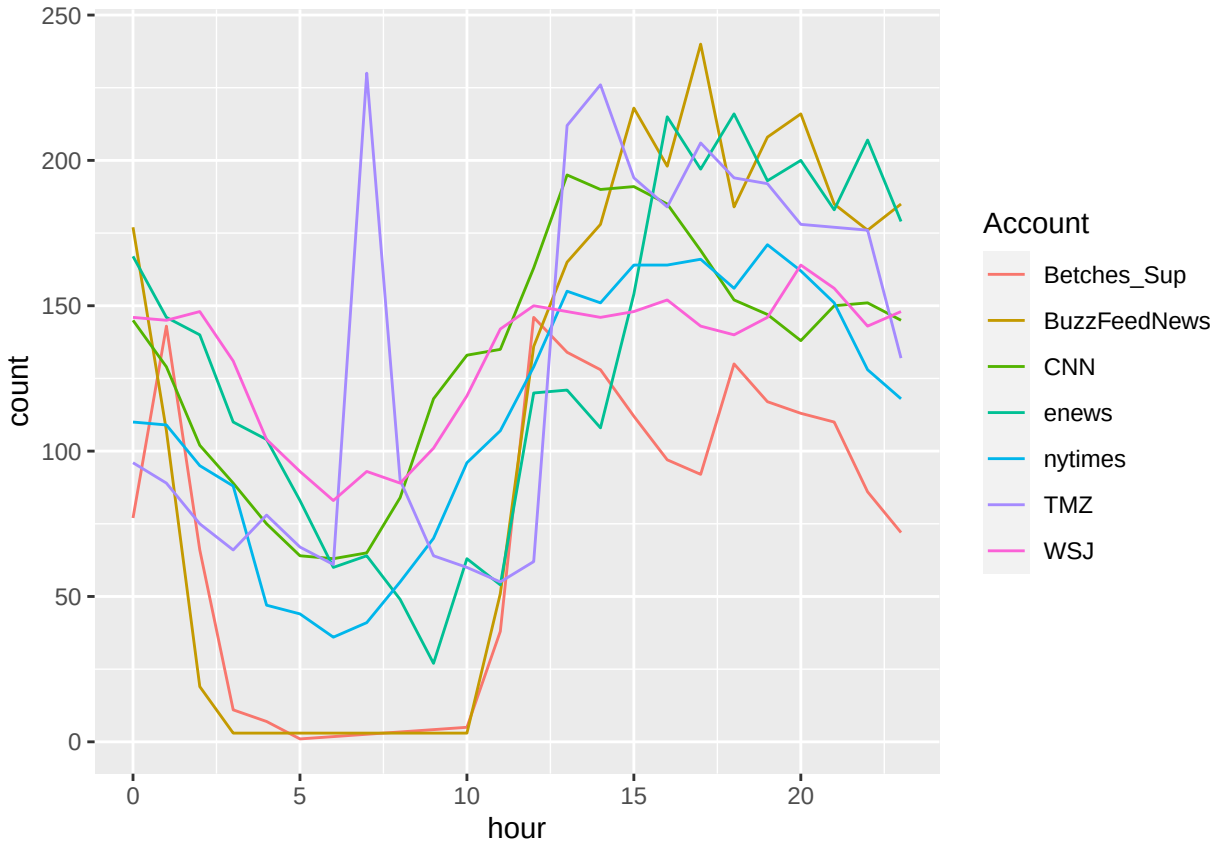
```
tweetsdate %>%
  filter(year == 2022) %>%
  group_by(Account,date)%>%
  mutate(count = n())%>%
  ggplot()+
  geom_line(aes(x=date, y = count, group=Account, color=Account)) +
  ggtitle("Tweets per Day by Account")
```



This confirms that betches_sup tweets significantly less frequently than the other accounts and shows that Buzzfeed also doesn't tweet as often as some. Since Betches and Buzzfeed don't tweet as often, I am looking at a longer time frame for these accounts. This is good to know for the topic modeling since their tweets may contains some topics that are not included in the other accounts, but is not a huge concern for the sentiment analysis since the overall vibe of how topics are discussed for each account should be fairly consistent.

I also can look at what time of day each account is tweeting.

```
tweetsdate %>%
  group_by(Account,hour)%>%
  mutate(count = n())%>%
  ggplot()+
  geom_line(aes(x=hour, y = count, group=Account, color=Account))
```



Each account follow a fairly similar pattern with less tweets in the morning and picking up as the day goes on. While it is interesting to see that TMZ has a spike around 6am, this probably doesn't have a huge impact on the deeper analysis.

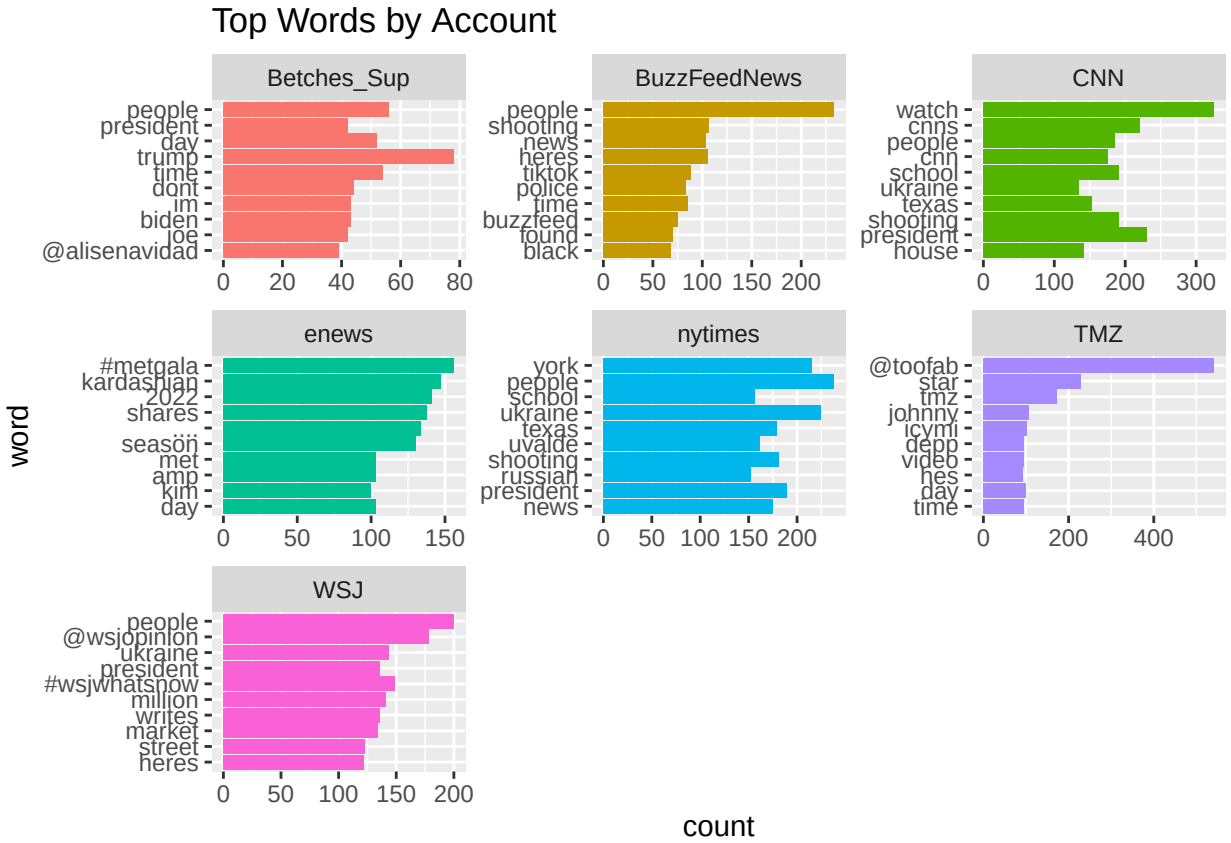
Data Cleaning

Now that I have pulled the data, let's start to clean it. I can start by tokenizing the text. This will make all text lowercase, remove punctuation and break the text down so that there is only one token (meaningful unit of text) per row. I also have added a parameter to strip the URL off the end of the tweet.

```
#tokenize
tweets_word <- tweets %>%
  unnest_tweets(output = word, input = text, strip_url = TRUE) %>%
  anti_join(stop_words)
```

Now that I have tokenized, I are able to look at the top used words across all tweets and by each different account. This will help us see if there are any words that I should add to the stop list.

```
# top words for each account
tweets_word %>%
  group_by(Account,word) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  slice_max(count,n = 10) %>%
  ggplot() +
  geom_col(aes(x = count, y = reorder(word,count), fill=Account),show.legend = FALSE) +
  facet_wrap(~ Account, scales = "free") +
  ggtitle("Top Words by Account") +
  ylab("word")
```

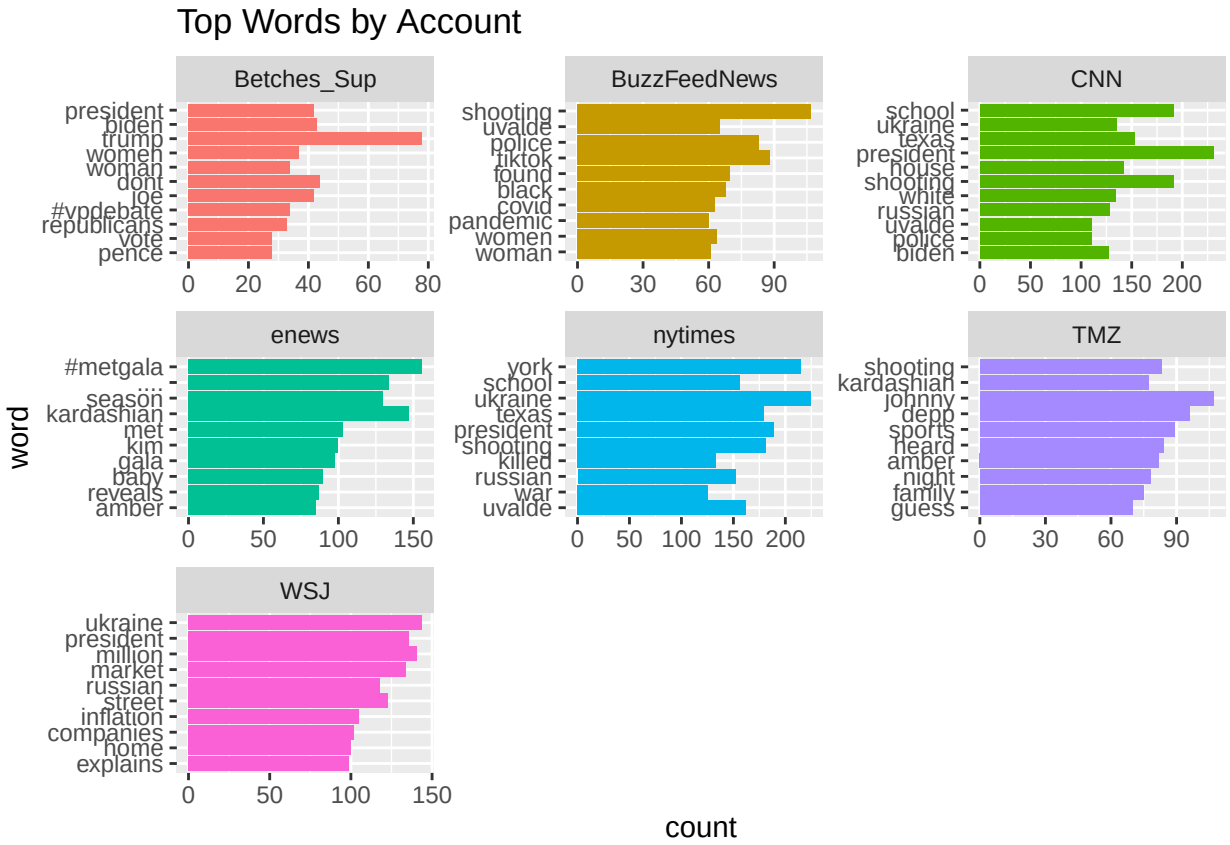


After looking at the top words for each account, I am able to add some additional words to the stop list based on what doesn't seem to be adding meaning. For example, people is in the top words for many accounts, therefore it is not adding any differentiating value. I also removed the names of the accounts, any @'s, and words like "breaking", "news", "watch", or "video" that would be common news headlines across all types of accounts.

```
# adding additional stop words
new_stops <- tibble(word = c("t.co", "https", "people", "it's", "@wsjopinion", "#wsjwhatsnow", "wsj",
                             "cnn", "cnns", "buzzfeed", "betches", "nyt", "nytimes", "enews", "breaking
                             "news", "amp", "day", "tmz", "@toofab", "icymi", "video", "watch", "time",
                             "2022", "@alisenavidad", "star", "u.s", "writes", "hes", "shes", "de",
                             "im", "shares", "heres"), lexicon = "nicole")

full_stops <- stop_words %>%
  bind_rows(new_stops)

tweets_word <- tweets_word %>%
  anti_join(full_stops)
```

Now, let's relook at the top words for each account and see if it looks better or if there are any additional words that need to be added.

```
# top words for each account pt 2
tweets_word %>%
  group_by(Account,word) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  slice_max(count,n = 10) %>%
  ggplot() +
  geom_col(aes(x = count, y = reorder(word,count), fill=Account),show.legend = FALSE) +
  facet_wrap(~ Account, scales = "free") +
  ggtitle("Top Words by Account") +
  ylab("word")
```



Topic Modeling

Now I need to structure the data into a document-term matrix so that I can complete a topic analysis.

```
# create document term matrix for topic modeling
tweets_word_count <- tweets_word %>%
  count(Account,word)

word_dtm <- tweets_word_count %>%
  cast_dtm(document = Account, term = word, value = n)
```

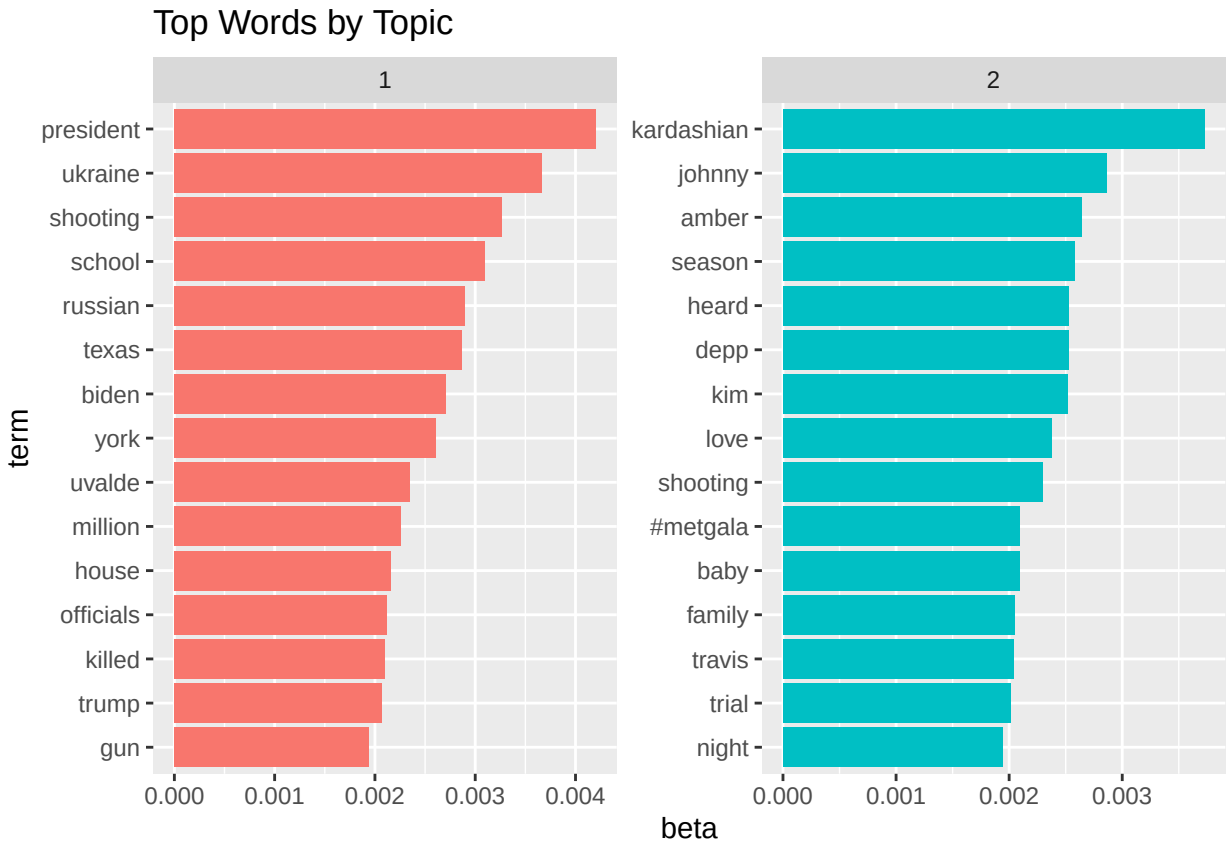
Two Topics: Hard News vs Soft News

Let's start of by creating two topics to see if the model can tell the difference between hard news and soft news sources. I am expecting to see WSJ, NYTimes, and CNN in one group with enews and TMZ in the other group. Assuming the groups end up that way, what I want to be looking at if Betches Sup and Buzzfeed News are a mix of hard news and soft news or if they lean more in one direction or the other.

To start off, I need to make the topics.

```
# 2 topics
lda_word_2 <-  LDA(word_dtm,k = 2, control = list(seed = 123))

# topic word probabilities
```

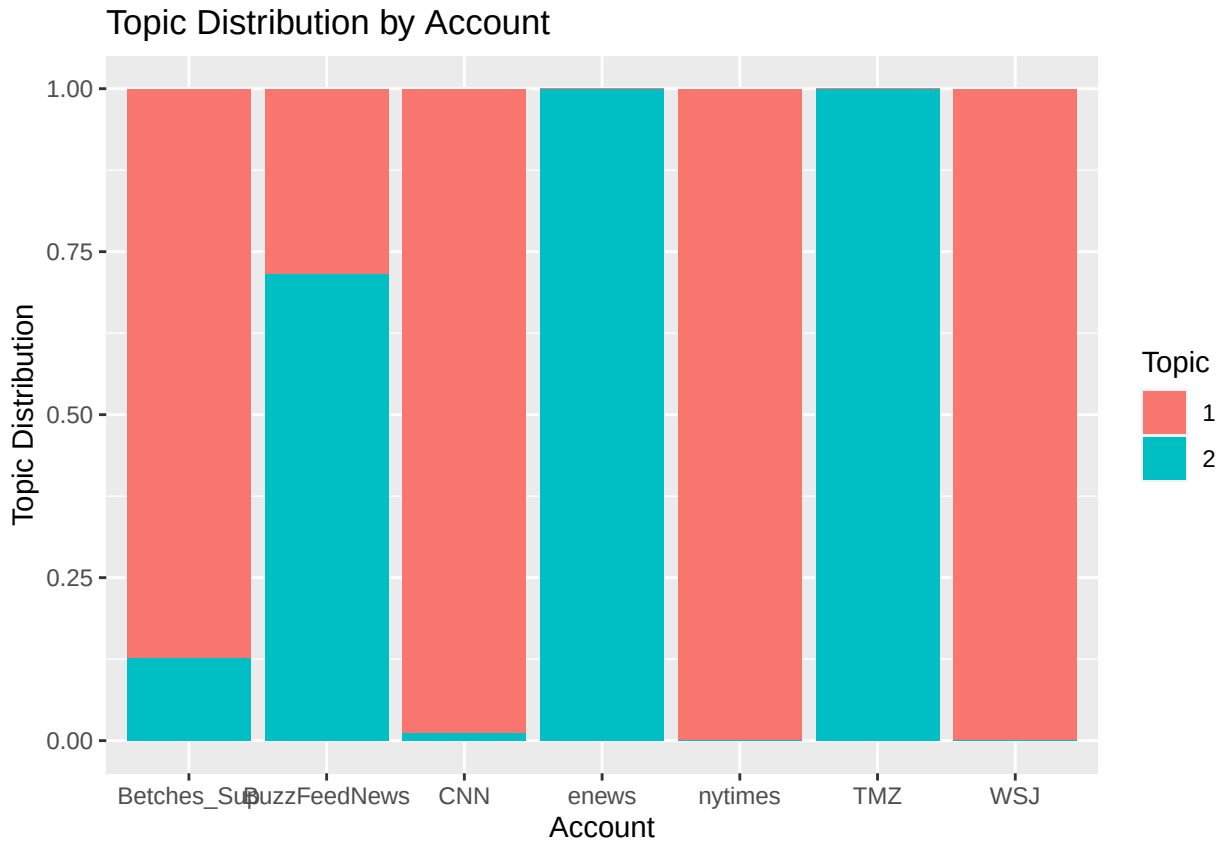
```r
beta_word_2 <- tidy(lda_word_2,matrix = "beta")

# most common words in each topic
top_terms10 <- beta_word_2 %>%
  group_by(topic) %>%
  slice_max(beta,n = 15) %>%
  ungroup() %>%
  arrange(topic,-beta)

# graph most common words in each topic
top_terms10 %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(x = beta, y = term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free")+
  scale_y_reordered()+
  ggtitle("Top Words by Topic")
```



When looking at these two graphs, I are able to see that the topics came out exactly how I were expecting. Topic 1 are the hard news topics such as the war in Ukraine, school shootings, President Biden, and covid while Topic 2 contains soft news topics such as the Kardashians, the Johnny Depp trial, and the met gala.

Now I want to look at the breakdown of these topics for each account, paying special attention to Buzzfeed and Betches.

```r
# document topic probabilities
gamma_word_2 <- tidy(lda_word_2, matrix = "gamma")

# graph topic breakdown by account
gamma_word_2 %>%
  ggplot() +
  geom_col(aes(document,gamma,fill=factor(topic))) +
  scale_fill_discrete(name = "Topic") +
  ggtitle("Topic Distribution by Account") +
  ylab("Topic Distribution") +
  xlab("Account")
```



As predicted, CNN, NYTimes and WSJ are almost all hard news (Topic 1), while enews and TMZ are all soft news (Topic 2). We now can see that Buzzfeed is almost 75% soft news while Betches is 90% hard news.

News Stories as Topics (Token = Word)

Now, I want to create more topics and try to see how different stories talked about in the news are spread among the various sources. To do this, I tried running the two above blocks of code with various different K values, but each topic would only show up in one account. I had been hoping to see the same topic across multiple accounts to show different sources covering the same stories.

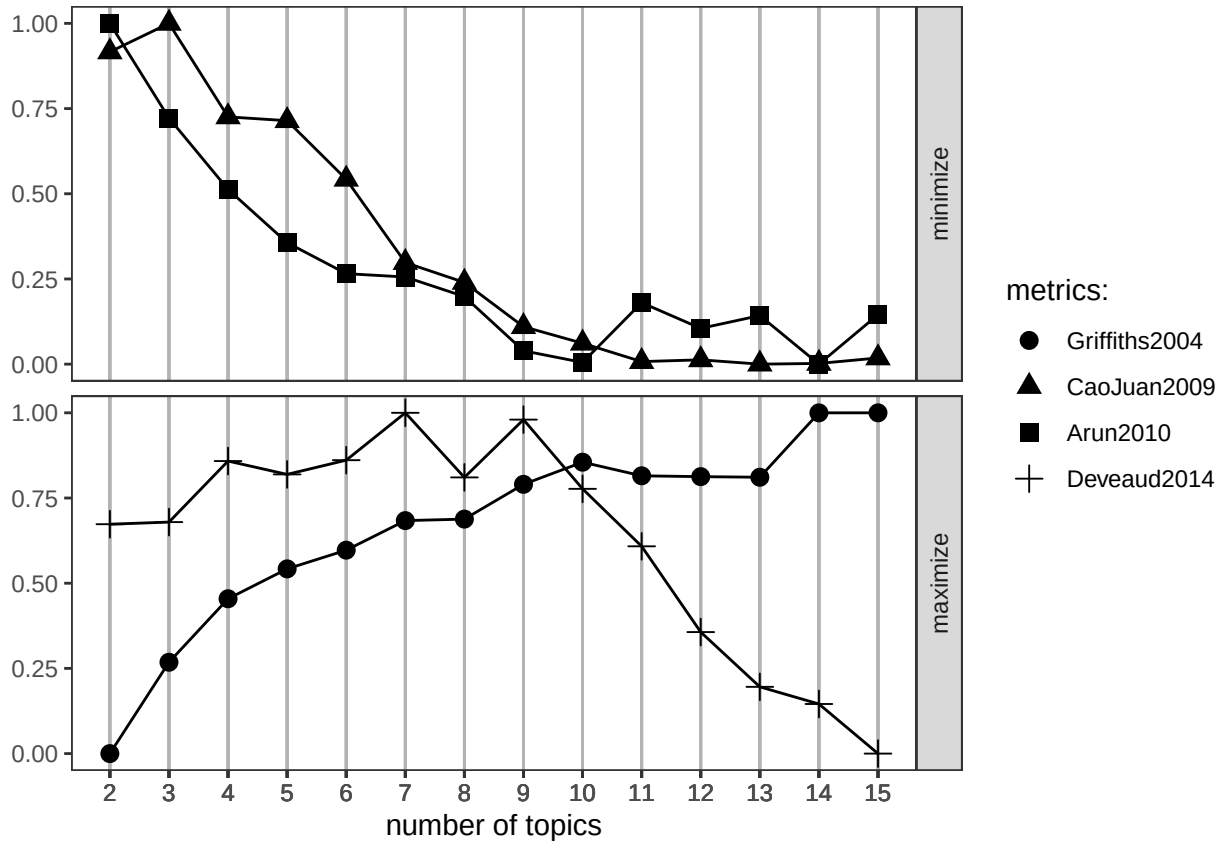
I found this function online to help find the optimal number of topics. We will run it trying K values from 2 to 15 in increments of 1. For each number of topics, the function calculates 4 different scores. For the Griffiths2004 and CaoJuan2009, I am looking for what value K will minimize these values. For Arun2010 and

Deveaud2014, I am looking to maximize these values. (Source: https://cran.r-project.org/web/packages/ldatuning/vignettes/topics.html#references)

```
result <- FindTopicsNumber(
  word_dtm,
  topics = seq(from = 2, to = 15, by = 1),
  metrics = c("Griffiths2004", "CaoJuan2009", "Arun2010", "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 77),
  mc.cores = 2L,
  verbose = TRUE
)
```

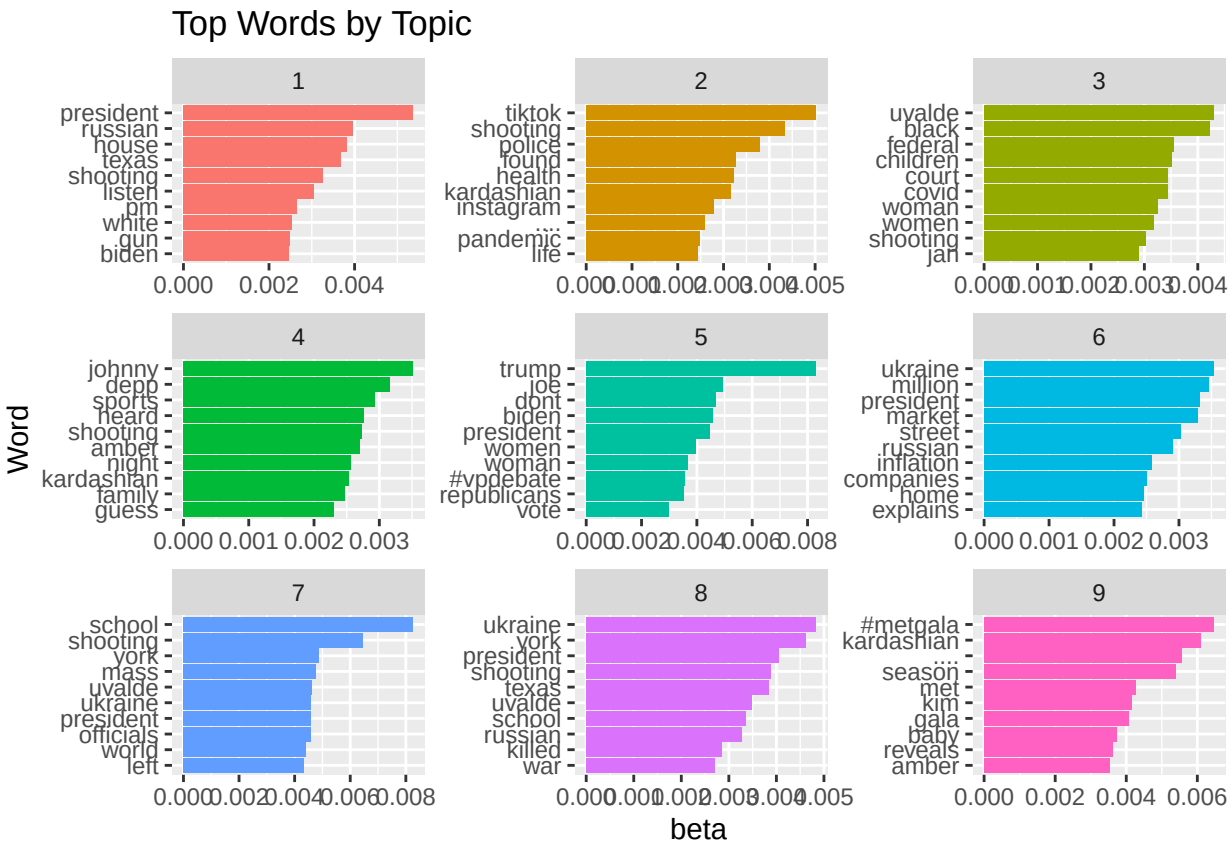
```
fit models... done.
calculate metrics:
  Griffiths2004... done.
  CaoJuan2009... done.
  Arun2010... done.
  Deveaud2014... done.
```

```
FindTopicsNumber_plot(result)
```



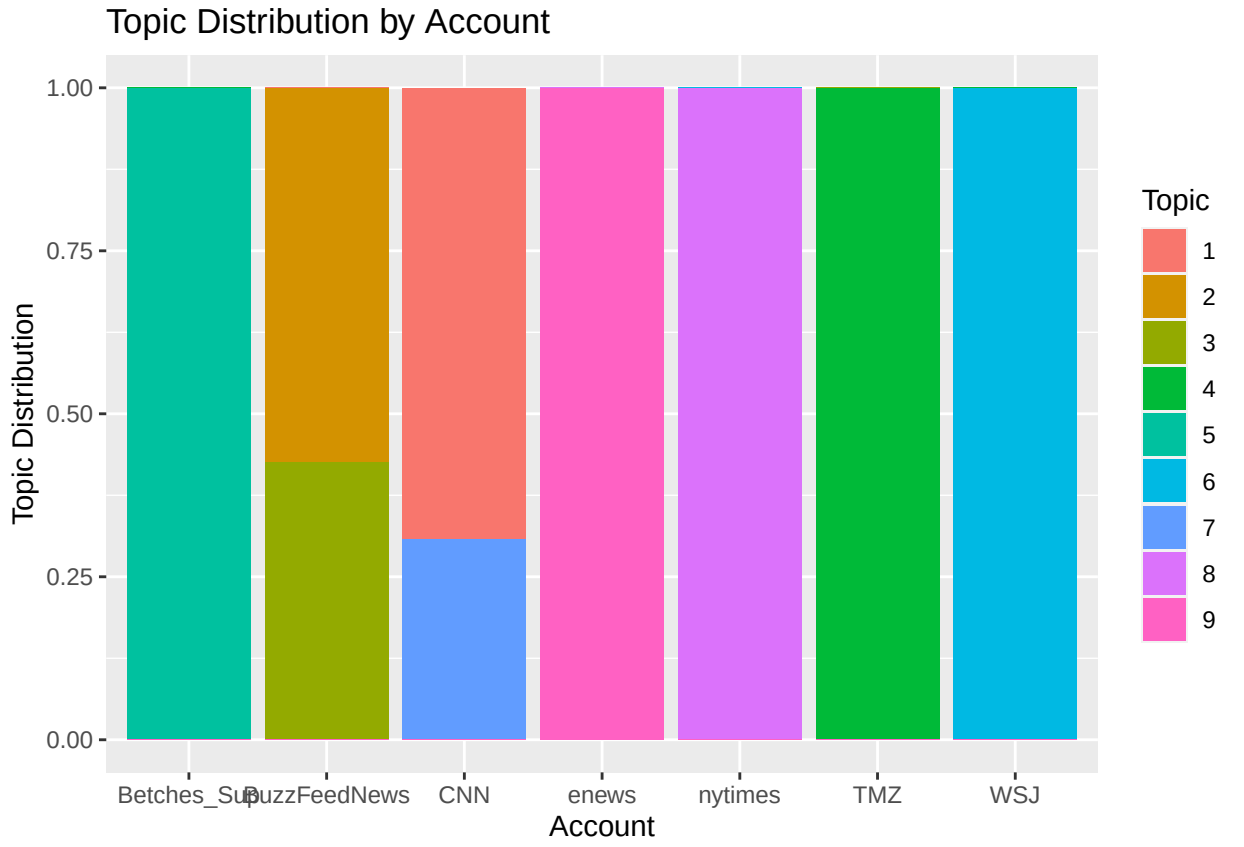
Based on the above graph, the optimal number of topics will be around 9. Let's try creating 9 topics.

```r
# 9 twitter accounts = 9 topics
lda_word_9 <-  LDA(word_dtm,k = 9, control = list(seed = 123))

# topic word probabilities
beta_word_9 <- tidy(lda_word_9,matrix = "beta")

# most common words in each topic
top_terms10 <- beta_word_9 %>%
  group_by(topic) %>%
  slice_max(beta,n = 10) %>%
  ungroup() %>%
  arrange(topic,-beta)

top_terms10 %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()+
  ggtitle("Top Words by Topic") +
  ylab("Word")
```



Now, let's see how these topics are spread out between the different documents.

```r
# document topic probabilities
gamma_word_9 <- tidy(lda_word_9, matrix = "gamma")

gamma_word_9 %>%
  ggplot() +
  geom_col(aes(document,gamma,fill=factor(topic))) +
  scale_fill_discrete(name = "Topic") +
  ggtitle("Topic Distribution by Account") +
  ylab("Topic Distribution") +
  xlab("Account")
```



Unfortunately, the topics still seem to be grouping by account. In order to get this to work the way I was hoping, I probably need to revisit my data cleaning and remove a lot more words that aren't adding meaning in order to narrow it down to just words that are related to key news topics. I also could look into using bigrams instead of words along with stemming or lemmatization to try and reduce the noise in my data by bringing words back to their root.

For the next step, let's try using Bigrams.

News Stories as Topics (Token = Bigram)

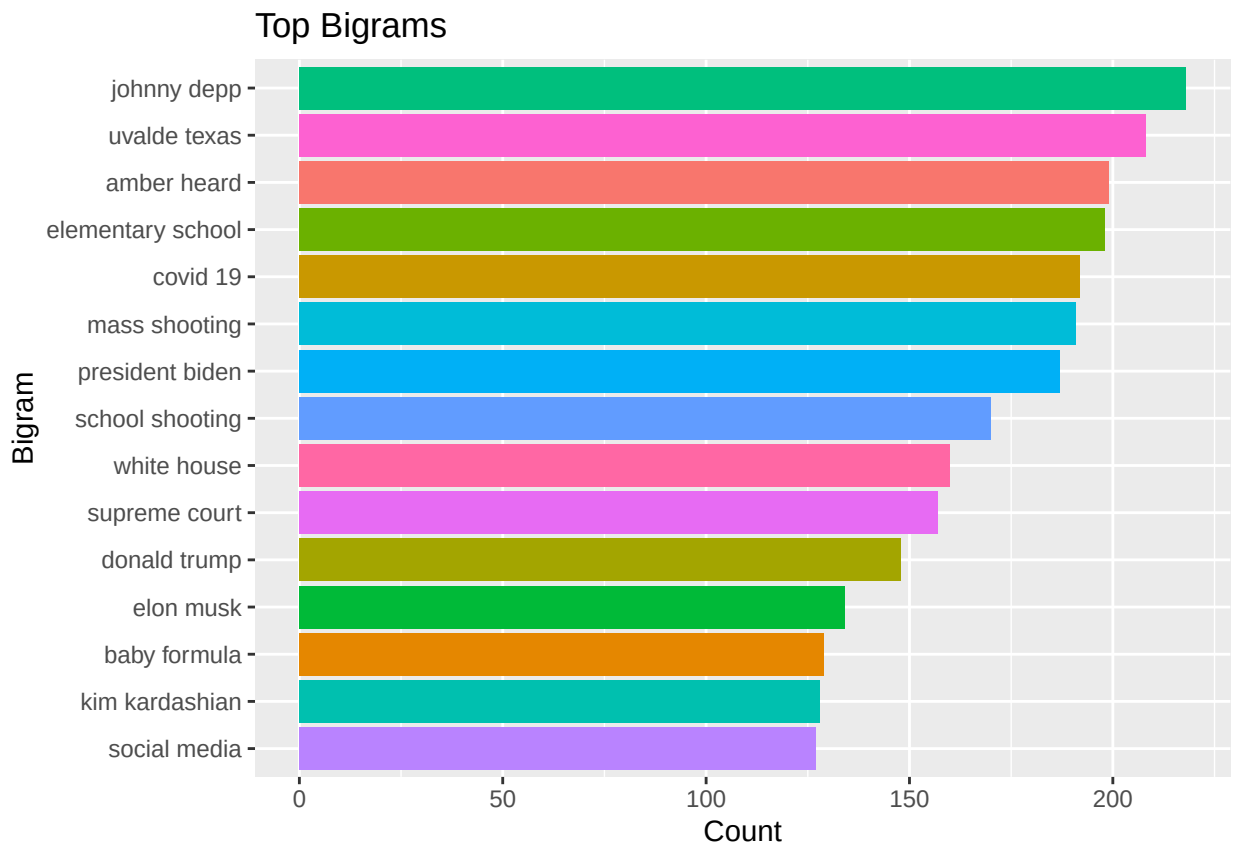
Let's organize the data into bigrams and retry modeling the various news stories as topics across our sources.

```r
# unnest into bigrams
tweets_bigram <- tweets %>%
```

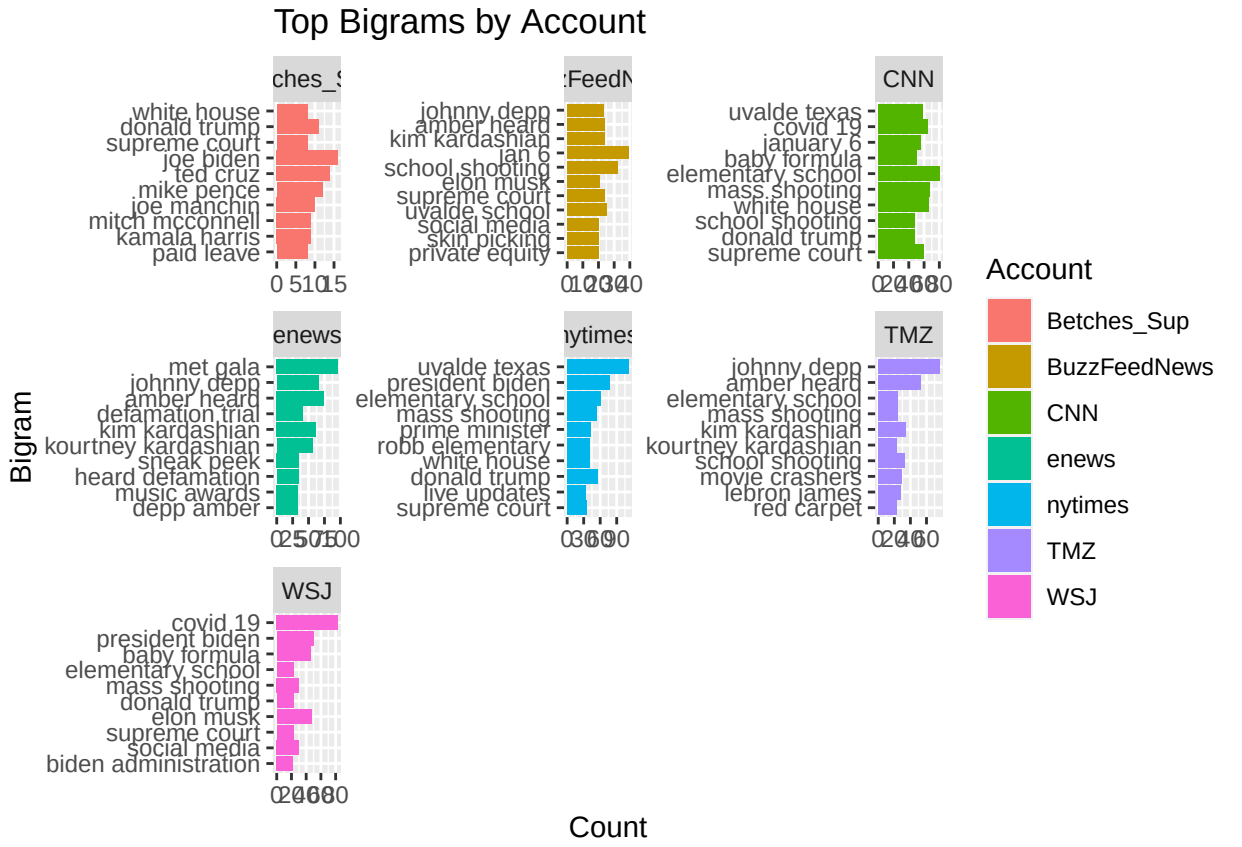
```
  unnest_tokens(output = bigram, input = text, token = "ngrams", n =2)

# separate the bigrams into two columns so that stop words can be removed
bigrams_separated <- tweets_bigram %>%
  separate(col = bigram, c("word1", "word2"), sep = " ")

#creating new stop list because the unnest_tokens function works slightly differently than the unnest_t
new_stops2 <- tibble(word = c("t.co", "https", "people", "it's", "wsjopinion", "wsjwhatsnow",
                              "wsj", "cnn", "cnns", "buzzfeed", "betches", "nyt", "nytimes",
                              "enews", "breaking", "news", "amp", "day", "tmz", "toofab",
                              "icymi", "video", "watch", "time", "2022", "alisenavidad",
                              "star", "u.s", "writes", "hes", "shes", "de", "im", "shares",
                              "heres", "new", "york", "times", "wall", "street", "journal",
                              "city", "front", "page"), lexicon = "nicole")

full_stops2 <- stop_words %>%
  bind_rows(new_stops2)

# removing stop words and filtering out blanks due to words being removed
bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% full_stops2$word) %>%
  filter(!word2 %in% full_stops2$word) %>%
  filter(word1 != "NA") %>%
  filter(word2 !="NA")

# recombining bigrams and making sure we removed all blanks
bigrams2 <- bigrams_filtered %>%
  unite(bigrams, word1,word2, sep = " ") %>%
  filter(bigrams != "NA NA")
```

Now that we have created our bigrams, lets look at our overall top bigrams and top bigrams by account in order to have a better idea of what our data looks like now.

```
# top bigrams in all tweets
bigrams2 %>%
  count(bigrams, sort = T) %>%
  slice_max(n,n = 15) %>%
  ggplot() +
  geom_col(aes(x = n, y = reorder(bigrams,n), fill=bigrams),show.legend = FALSE) +
  # facet_wrap(~Account, scales = "free") +
  ggtitle("Top Bigrams") +
  ylab("Bigram")+
  xlab("Count")
```




```
# top bigrams by account
bigrams2 %>%
  group_by(Account,bigrams) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  slice_max(count,n = 10) %>%
  ggplot() +
  geom_col(aes(x = count, y = reorder(bigrams,count), fill=Account)) +
  facet_wrap(~ Account, scales = "free") +
  ggtitle("Top Bigrams by Account") +
  ylab("Bigram")+
  xlab("Count")
```

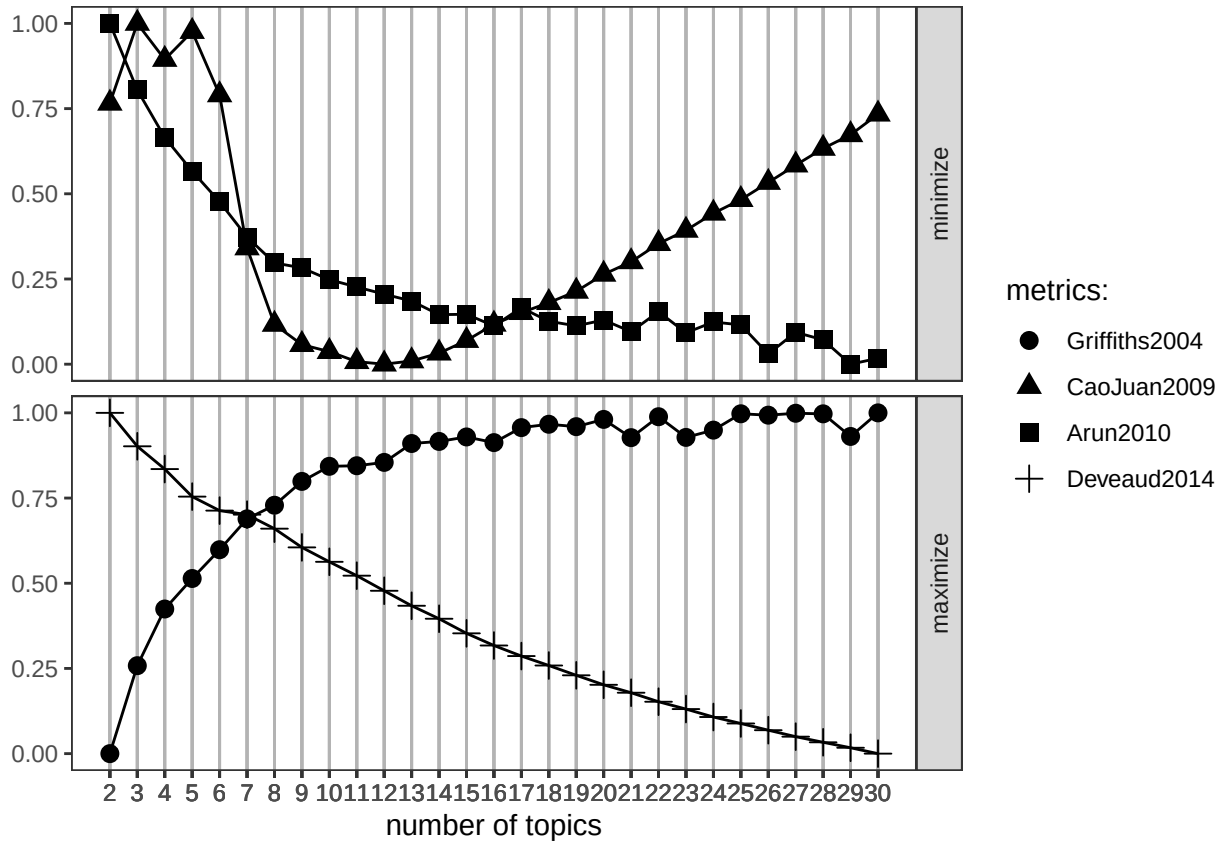


Now I need to create a document-term matrix to prep the bigram dataset for topic modeling.

```
# create DTM
tweets_bigram_count <- bigrams2 %>%
  count(Account,bigrams)

bigram_dtm <- tweets_bigram_count %>%
  cast_dtm(document = Account, term = bigrams, value = n)

# run topic number function to final optimal numbner of topic
result <- FindTopicsNumber(
  bigram_dtm,
  topics = seq(from = 2, to = 30, by = 1),
  metrics = c("Griffiths2004", "CaoJuan2009", "Arun2010", "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 77),
  mc.cores = 2L,
  verbose = TRUE
)
```

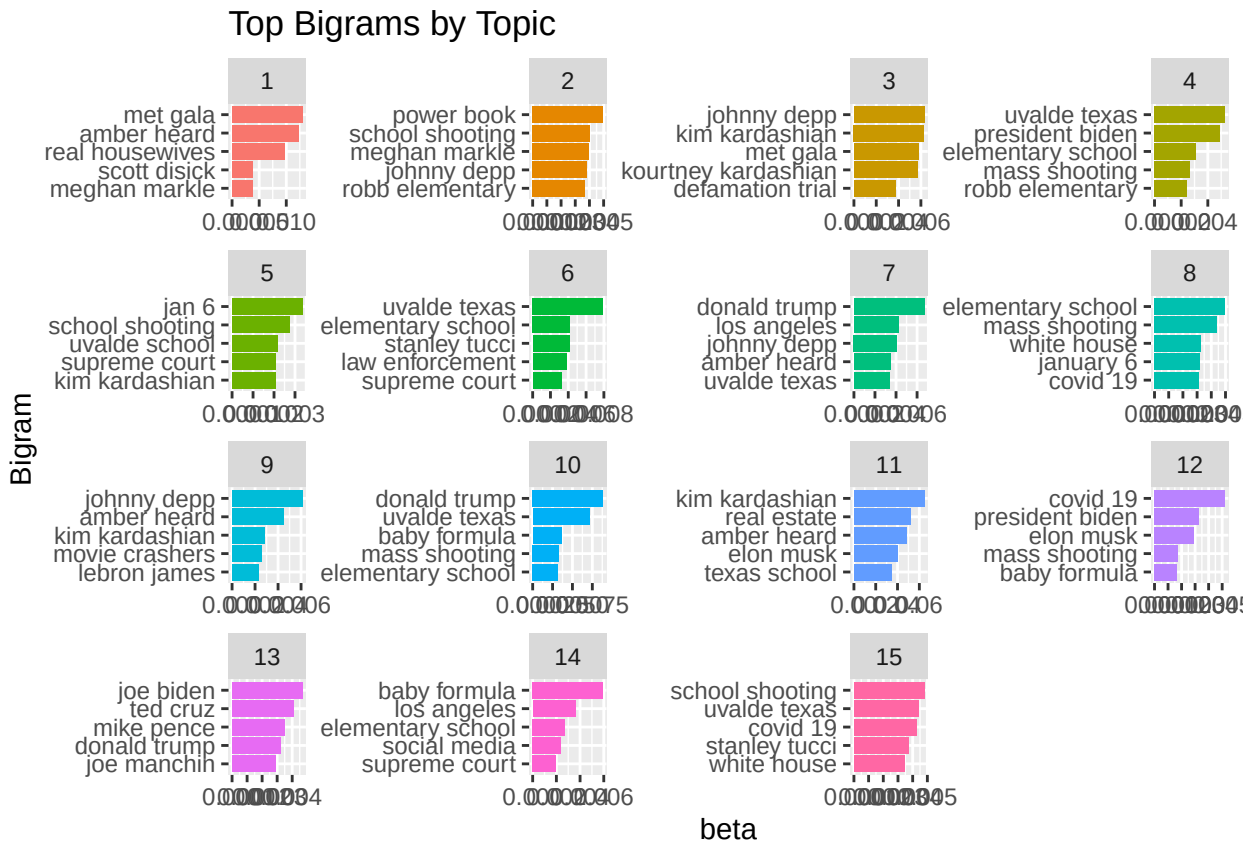
```
fit models... done.
calculate metrics:
  Griffiths2004... done.
  CaoJuan2009... done.
  Arun2010... done.
  Deveaud2014... done.
```

```
FindTopicsNumber_plot(result)
```



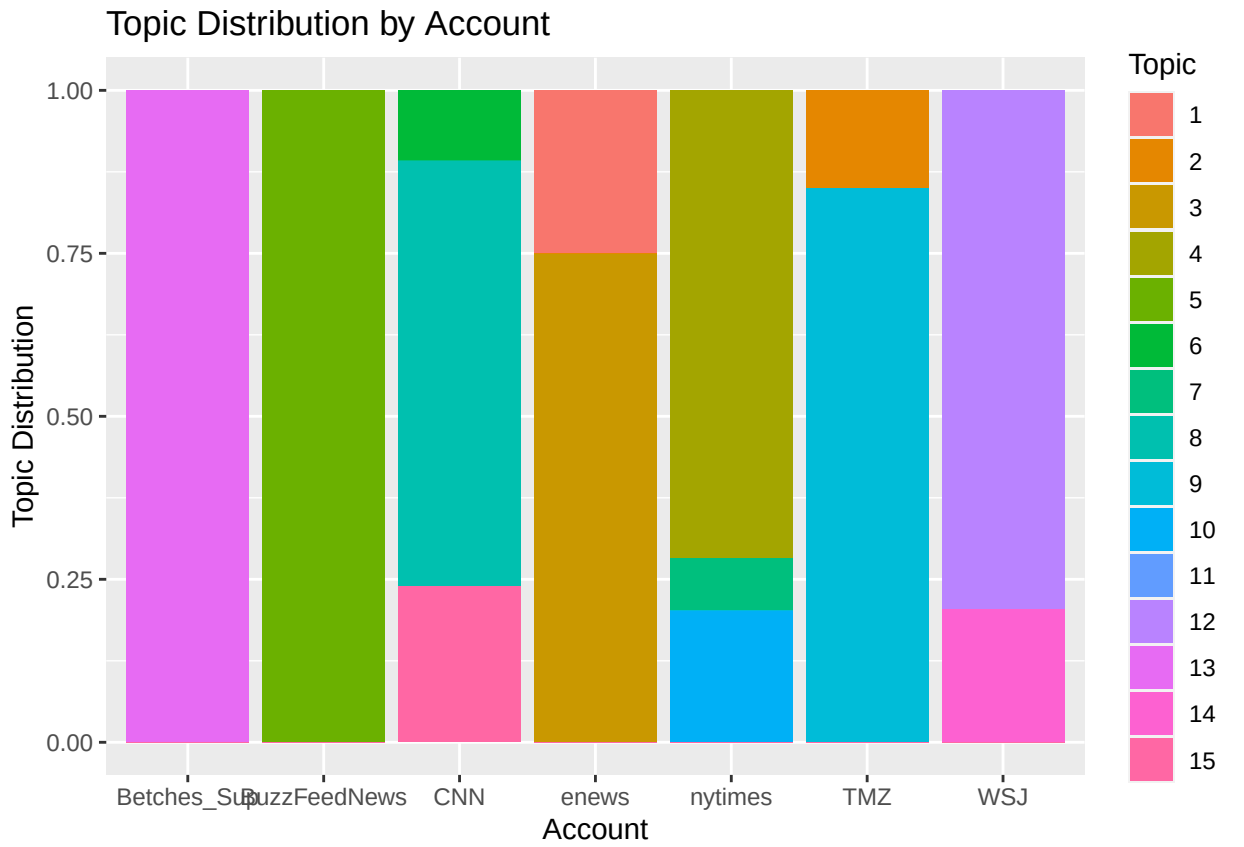
Based on this plot, I am going to start by trying 15 topics.

```
# 15 topics
lda_bigram_15 <-  LDA(bigram_dtm,k = 15, control = list(seed = 123))

# topic word probabilities
beta_bigram_15 <- tidy(lda_bigram_15,matrix = "beta")

# most common words in each topic
top_terms15 <- beta_bigram_15 %>%
  group_by(topic) %>%
  slice_max(beta,n = 5) %>%
  ungroup() %>%
  arrange(topic,-beta)

top_terms15 %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()+
  ggtitle("Top Bigrams by Topic") +
  ylab("Bigram")
```



Based on this, we can see that there is a lot of overlap on the top bigrams in each topic. This probably means that our topics are still split up based on the news source rather than grouping only by news topic.

```
# document topic probabilities
gamma_bigram_15 <- tidy(lda_bigram_15, matrix = "gamma")



gamma_bigram_15 %>%
  ggplot() +
  geom_col(aes(document,gamma,fill=factor(topic))) +
  scale_fill_discrete(name = "Topic") +
  ggtitle("Topic Distribution by Account") +
  ylab("Topic Distribution") +
  xlab("Account")
```



Once again, the topics still seem to be grouping by account. This shows that I probably need to clean my data more so that we are just looking at the key words for each news topic in order to get the results that I was hoping for.

For now, I am going to leave that as a next step that can be completed at a later time since that can be very time consuming but it would closely follow the concepts that have already been outlined above. In order to continue this analysis, I would continue to add words to my stop words list and remove then from the dataset. I would then create my document-term matrix, look for the optimal number of topics and run a topic model with the hopes of eventually having removed enough stop words to see the same topic appearing across multiple news sources. This would show that the topics are representing news stories such as the Ukraine war or the Johnny Depp v. Amber Heard defamation trial.

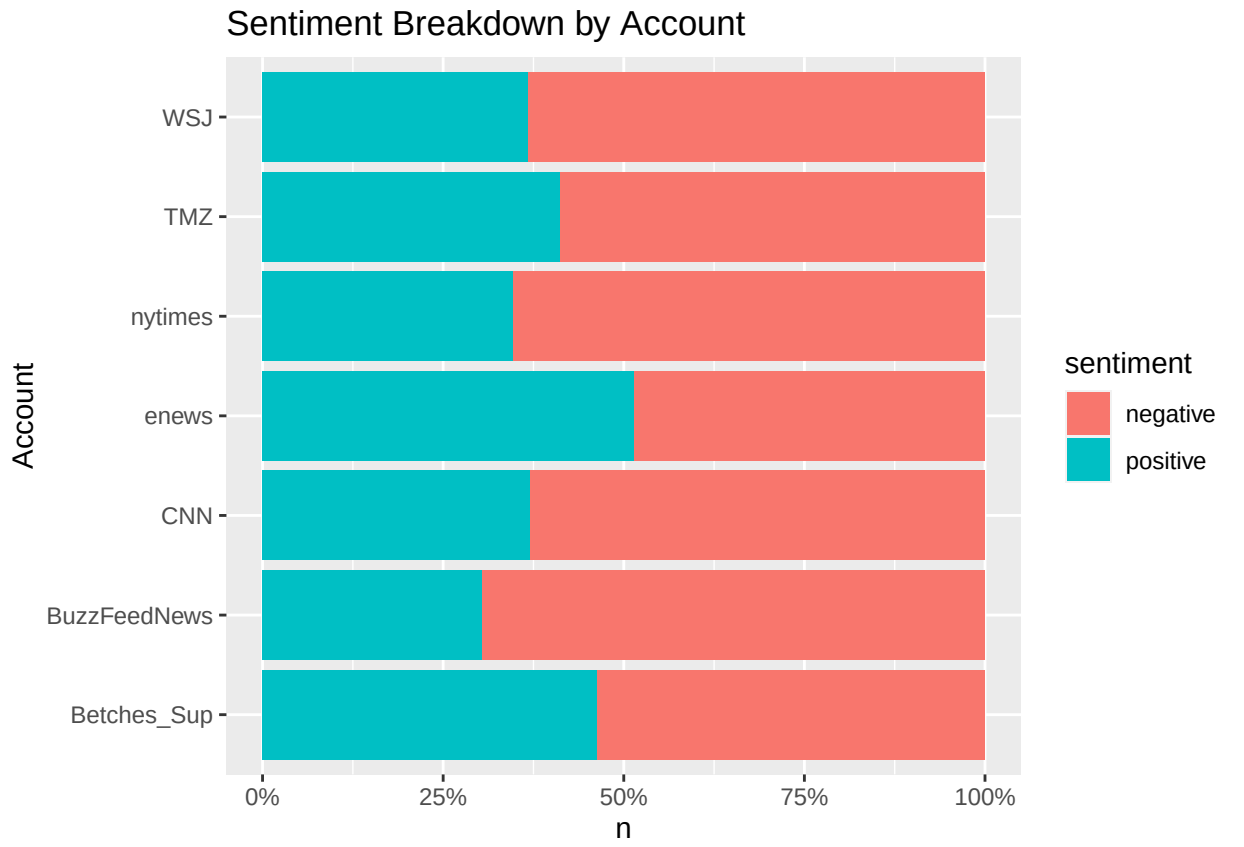
Tweet Sentiments

Now, I want to try and get an understanding of the sentiment of each account. We know that WSJ, NYT, CNN, Betches Sup discuss similar topics to each other and the same goes for enews, TMZ, and Buzzfeed. From here on out, when I refer to hard news accounts I will be speaking of WSJ, NYT, CNN, Betches Sup while soft news accounts will be enews, TMZ, and Buzzfeed. We can compare the sentiments between the various accounts to look at how each account frames the topic. For example, are any hard news accounts framing the same stories in a more negative way compared to other hard news accounts?

First, I can look at the percent of positive vs negative terms used by each account. I will start of by using the bing lexicon to mark each word as positive or negative.

```
tweets_word %>%
  inner_join(get_sentiments("bing")) %>%
  count(Account, sentiment) %>%
```

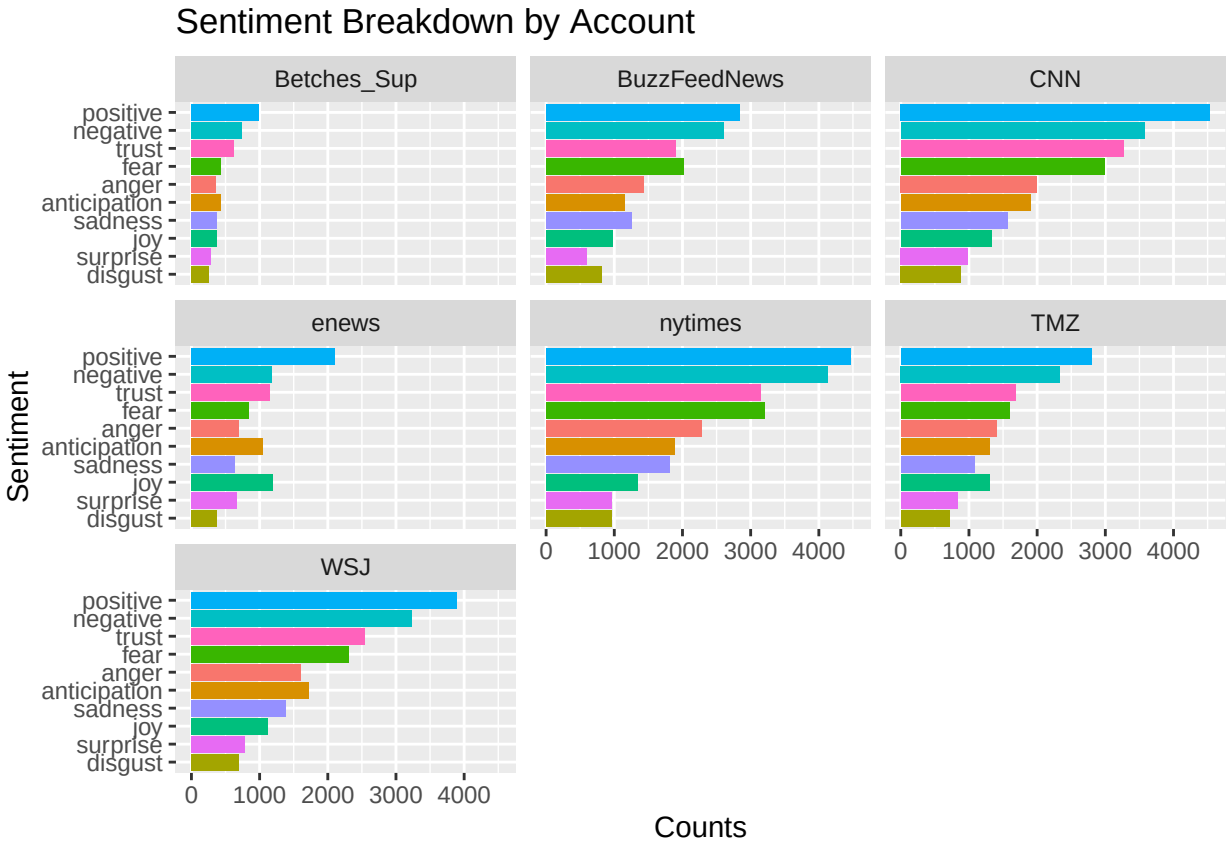
```
  ggplot(aes(y=Account,x=n,fill=sentiment)) +
  geom_col(position="fill") +
  scale_x_continuous(labels = scales::percent)+
  ggtitle("Sentiment Breakdown by Account")
```



Unfortunately, all of the accounts are majority negative, with the exception of enews which is 50:50. Surpisingly, Buzzfeed news is the most negative which is interesting because I typically think of Buzzfeed as a somewhat lighthearted website. It also is interesting that enews is the most positive because celebrity news sites tend to have a reputation for being negative, but this may show that enews is sharing positive celebrity news and not just gossip. For our hard news accounts: WSJ, NYT and CNN all hover around 40% positive, while Betches looks around 47% positive. I think it makes sense that Betches is slightly more positive than the others since they tend to simplify or make jokes about the news.

Now, I can break down the sentiment into more different emotions by using a different sentiment lexicon. Using the nrc lexicon will output emotions such as trust, fear, anger, anticipation, sadness, etc in addition to positive and negative.

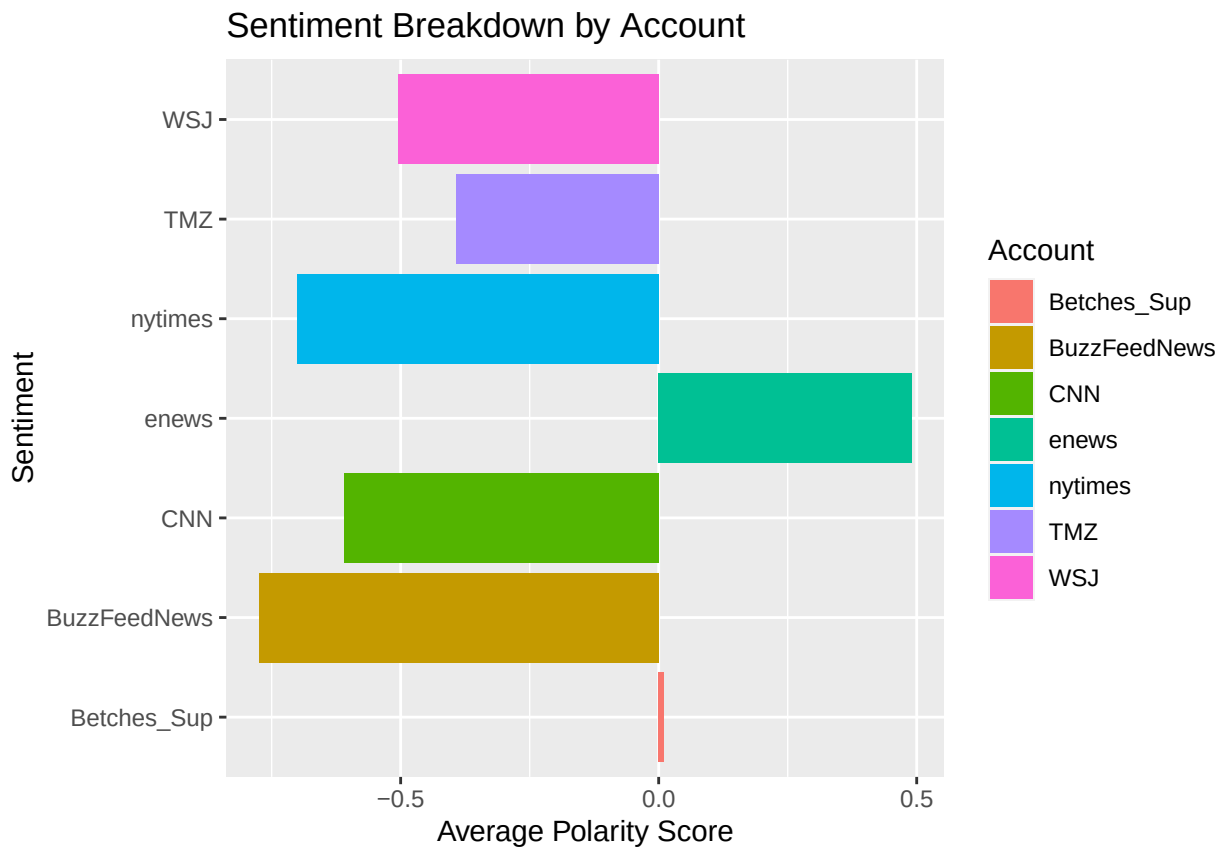
```
tweets_word %>%
  inner_join(get_sentiments("nrc")) %>%
  count(Account, sentiment) %>%
  ggplot() +
  geom_col(aes(x = n, y = reorder(sentiment,n), fill = sentiment), show.legend = F) +
  facet_wrap(~Account) +
  ggtitle("Sentiment Breakdown by Account") +
  ylab("Sentiment") +
  xlab("Counts")
```



The first thing that is interesting here is that our top emotion appears to be positive, which is opposite of the last visual, but when you look at the emotions that follow it, more seem to be on the negative side. It also is interesting how TMZ and enews have little spikes for joy, while buzzfeed and nytimes have little spikes for fear.

I also can use the afinn sentiment to get a polarity score ranging from -5 to 5 for each word. I then can take the average of these scores for each account to see the average sentiment polarity for each account.

```
tweets_word %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(Account) %>%
  summarize(score = mean(value)) %>%
  ggplot(aes(y=Account,x=score,fill=Account)) +
  geom_col() +
  ggtitle("Sentiment Breakdown by Account") +
  ylab("Sentiment") +
  xlab("Average Polarity Score")
```



Once again, enews is the most positive account. What is interesting here is that in our negative/positive sentiment analysis, enews was 50:50 but when we add in weighted values for each word, enews becomes overwhelmingly positive. Buzzfeed once again is the most negative account, followed by the nytimes.It is interesting to see that betches_sup averaged out around zero, despite it being most similar in topic to nytimes, CNN, and WSJ which all three were in the bottom 4.

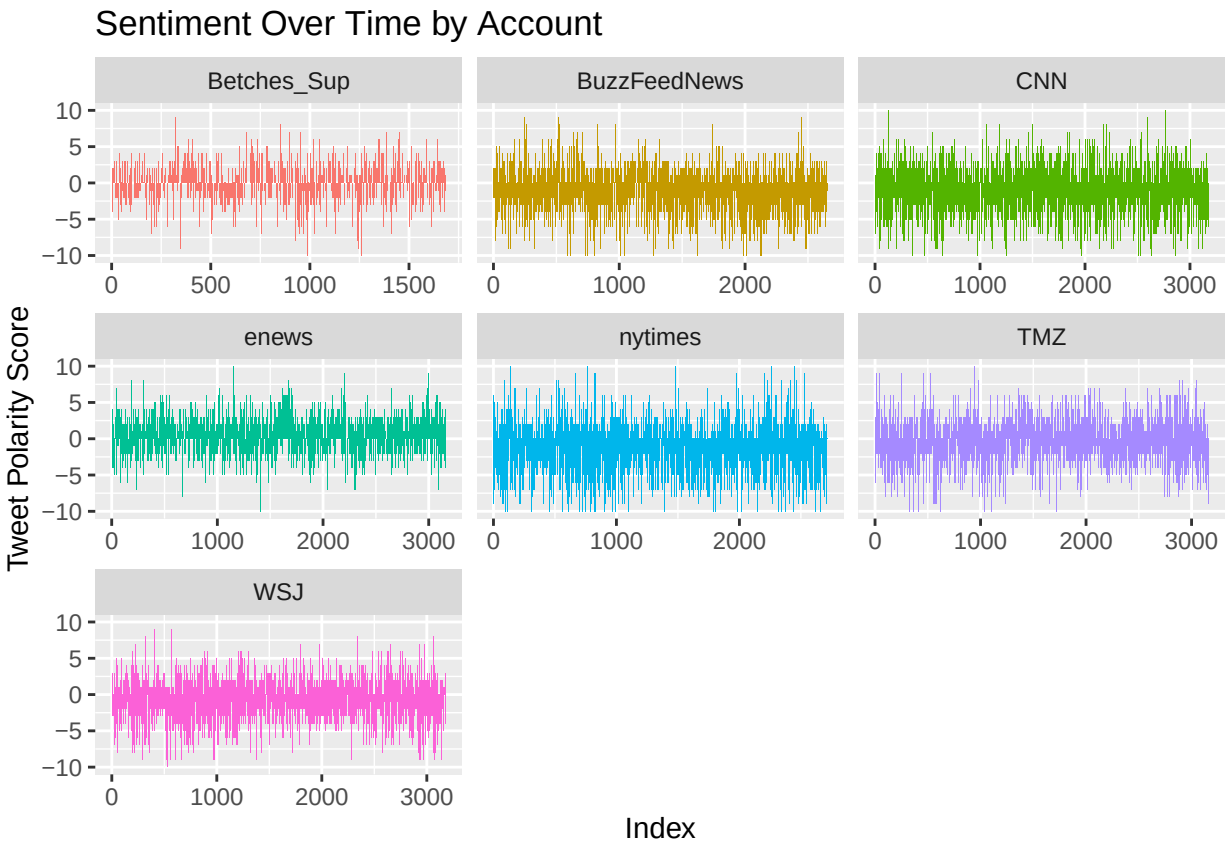
Lastly, we can look at the sentiment over time and see if there have been any trends or changes. First, we need to set up an index to use as our time series. Here I will do this by putting the tweets in date order for each account and then numbering them in order. This will create an index for the tweet number.

```
tweets_word_index <- tweetsdate %>%
  arrange(created_at)%>% # make sure tweets are in date order
  group_by(Account) %>%
  mutate(index = row_number(Account)) %>% # create index column with the row number count within each A
  ungroup() %>%
  unnest_tweets(output = word, input = text, strip_url = TRUE) %>%
  anti_join(full_stops)
```

Now that I have made my index, I can graph the average sentiment score for each tweet across the index to see if there are any trends in sentiment.

```
tweets_word_index %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(Account, index) %>%
  summarize(score = sum(value)) %>%
  ggplot() +
```

```
geom_col(aes(x=index, y=score, fill = Account), show.legend = F) +
facet_wrap(~Account, scales = "free_x") +
ylim(-10,10) +
ggtitle("Sentiment Over Time by Account") +
ylab("Tweet Polarity Score") +
xlab("Index")
```



There are no clear patterns here but still interesting to look at. A next step could be to index by the date and get average sentiment polarity score for each day per account. Graphing this would allow you to see if accounts had positive and negative sentiments on the same day.

Conclusions

1. How similar are twitter accounts of various news outlets?

While the models were able to distinguish between the hard and soft news sources, I haven't yet been able to model news stories as topics to the various accounts. It was interesting to see that betches sup was classified as mostly hard news since most of their content is light-hearted reactions and interpretations to hard news topics. The next steps here will be to continue to clean the data to see if it is possible to build topics that represent the various news stories being discussed to see if certain sources talk more about one story than another. To continue cleaning the data, I will want to remove more words until just key words about news topics remain.

2. For news outlets that report on similar topics, is the sentiment also the same?

Unfortunately, all of the accounts are majority negative, with the exception of enews. This was surprising because often times celebrity news outlets have a reputation for being negative, but for each sentiment analysis completed, enews appeared positive. This shows that enews may be sharing celebrity gossip in a positive way and not only spreading dirty gossip. This was in contrast to buzzfeed and TMZ who both tweeted about similar topics to enews, but had negative sentiments. Buzzfeed news was the most negative which is interesting because I typically think of Buzzfeed as a somewhat lighthearted website and would have expected the most negative to be TMZ. For the hard news accounts, it is not surprising that betches_sup was the most positive out the group but it was surprising how different betches_sup's average polarity score was from the others.