Text Analysis: Dow Jones 10-K Item 1A. Risk Factors

```
# remove error messages in docuemnt construction
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
```

11/19/2023

Overview

Is the American economy heading for disaster or are things better than ever? There's no consensus. Macroeconomic trends are playing out before us:

- New normals: interpersonal behavior, economic activity, and supply chains in the aftermath of a COVID pandemic
- Decoupling of China and America as trade partners, anti-American axis forming between China, Iran, North Korea, and Russia
- Increasing conflict between regional rivals
- The end of a decade of easy money
- Banking: regional collapses, interest rate risk, flighty depositors
- Commercial real estate demand at historical lows, residential real estate prices at all time highs
- Inflation taming but still above historical levels, cooling hiring activity
- Politically divided country on the eve of an election, human mimetic behavior exploited and exasperated through internet and anonymity
- Changing climate
- The rise of AI and WEB3 as tools to improve or harm our world

While we can't predict the future of the American economy, we can assess the forward-looking Risk Factors for the 30 Dow Jones companies that represent it. We will attempt to peel back the rosy surface layer and understand the subtext. What information might have they let slip? How can we use the information to our benefit as customers and investors?

Contents

10-K Text Data Acquisition & Setup

To answer these questions, we will utilize the '1A. Risk Factor' sections the two most recent 10-Ks for all 30 Dow Companies. Fiscal years can be different from company to company, so there are 10-Ks from as recent as Q3 2023 to as far back as Q3 2021.

Each row represents a 10-K document. In addition to the risk factor text for each 10-K, the data contains the 10-K URLs in 'source_url', the company ticker in 'Tickers' which will appear twice in the data set, the 'Date' representing the filing quarter, the company sector in 'Sectors', the 'Industry', 'New_or_Old' which has values N and O representing the newer or older 10-K document for any given company, and finally, 'Ticker_Date' which is a concatenation of 'Tickers' and 'Date' to created an easily readable and shorter unique key.

As many will be interested in the analysis solely, we have made the bulk of this section an appendix of sorts: Discussion on SEC Data Scraping. I wanted to include details in hopes it helps else quicken others' data acquisition process in the future.

Exploratory Data Analysis

To get our bearings, let's import and clean the Data

```
library(stringr)
library(tidymodels)
library(tidyverse)



dow_risk_factors <- read_csv("dow_risk_factors.csv")

# rename headers
names(dow_risk_factors)[names(dow_risk_factors) == 'Source URL'] <- 'source_url'
names(dow_risk_factors)[names(dow_risk_factors) == 'Item 1A Text'] <- 'text'



# clean: lower words, remove punctuation and extra spaces
dow_risk_factors_cleaned <- dow_risk_factors %>%
    mutate(text = tolower(text)) %>%  # lowercase
    mutate(text = str_replace_all(text, "[[:punct:]]", " ")) %>%
    mutate(text = str_replace_all(text, "\\s+", " ")) %>%
```

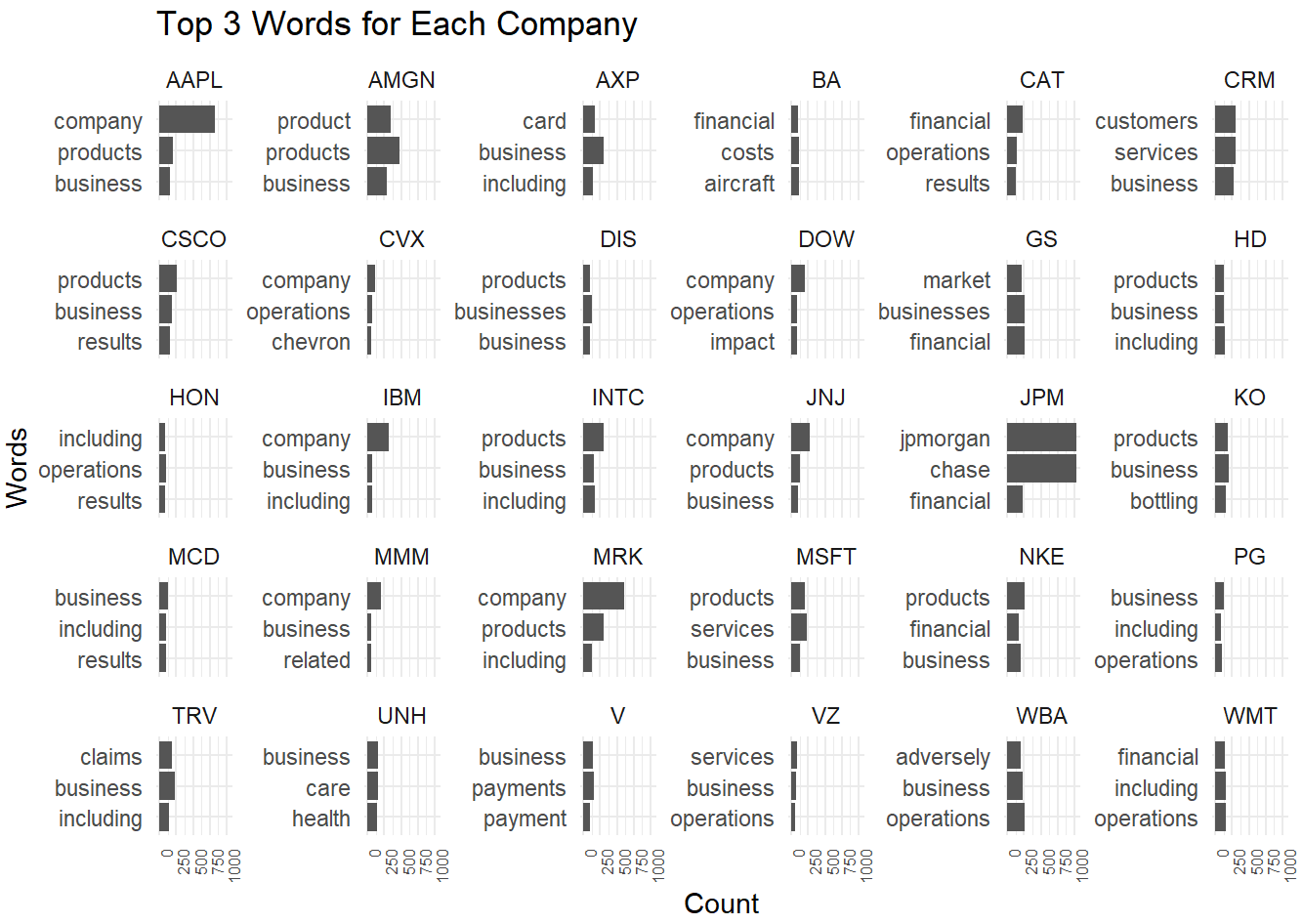
```
    mutate(ticker_date = str_c(Tickers, Date, sep = "_"))


# Some quick work to swap some text data that was swapped with lower row in the csv read-in.
row_IBM <- which(dow_risk_factors_cleaned$ticker_date == "IBM_2022-12-31")
row_AMGN <- which(dow_risk_factors_cleaned$ticker_date == "AMGN_2022-12-31")

# swap values
if(length(row_IBM) == 1 && length(row_AMGN) == 1) {
  temp <- dow_risk_factors_cleaned$text[row_IBM]
  dow_risk_factors_cleaned$text[row_IBM] <- dow_risk_factors_cleaned$text[row_AMGN]
  dow_risk_factors_cleaned$text[row_AMGN] <- temp
} else {
  warning("One or both rows not found")
}


dow_risk_factors_cleaned
```

```
# A tibble: 60 × 8
   source_url    text  Tickers Date       Sector Industry New_or_Old ticker_date
   <chr>         <chr> <chr>   <date>     <chr>  <chr>    <chr>      <chr>
 1 https://www.… "ite… AAPL    2023-09-30 Infor… Consume… N          AAPL_2023-…
 2 https://www.… "ite… MSFT    2023-06-30 Infor… Softwar… N          MSFT_2023-…
 3 https://www.… "ite… V       2023-09-30 Infor… Financi… N          V_2023-09-…
 4 https://www.… "ite… UNH     2022-12-31 Healt… Managed… N          UNH_2022-1…
 5 https://www.… "ite… WMT     2023-01-31 Consu… Retail   N          WMT_2023-0…
 6 https://www.… "ite… JPM     2022-12-31 Finan… Banking  N          JPM_2022-1…
 7 https://www.… "ite… PG      2023-06-30 Consu… Consume… N          PG_2023-06…
 8 https://www.… "ite… JNJ     2023-01-01 Healt… Pharmac… N          JNJ_2023-0…
 9 https://www.… "ite… HD      2023-01-29 Consu… Home Im… N          HD_2023-01…
10 https://www.… "ite… CVX     2022-12-31 Energy Oil and… N          CVX_2022-1…
# ℹ 50 more rows
```

Looking at the most common words per company, it is clear we will have to create a dictionary of corporate-speak stop words to further-remove.

```
library(tidytext)

dow_risk_tokens <- dow_risk_factors_cleaned %>%
  unnest_tokens(input = text,
                output = word) %>%
                anti_join(stop_words)

top_words_by_ticker <- dow_risk_tokens %>%
  group_by(Tickers) %>%
  count(word) %>%
  top_n(3, n) %>%
  ungroup() %>%
  arrange(Tickers, desc(n))
```

```
ggplot(top_words_by_ticker, aes(x = n, y = reorder(word, n))) +
  geom_col() +
  facet_wrap(~Tickers, scales = "free_y") +
  labs(title = "Top 3 Words for Each Company",
       x = "Count",
       y = "Words") +
  theme_minimal() +

  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 6))
```



Let's get a better sense of the top 15 words for each company.

```
top_words_by_ticker <- dow_risk_tokens %>%
  group_by(Tickers) %>%
  count(word) %>%
  top_n(15, n) %>%
  ungroup() %>%
  arrange(Tickers, desc(n))  # sorting by Tickers and then by n in descending order

top_words_by_ticker
```

```
# A tibble: 463 × 3
   Tickers word           n
   <chr>   <chr>      <int>
```

```
 1 AAPL    company      830
 2 AAPL    products     204
 3 AAPL    business     162
 4 AAPL    services     146
 5 AAPL    financial    126
 6 AAPL    operations   121
 7 AAPL    adversely    116
 8 AAPL    materially    85
 9 AAPL    results       85
10 AAPL    including     84
# ℹ 453 more rows
```

Let's now generate a unique list from the top 15 most used words for each company to serve as our general list of stop words. We will also 'save' words by hand picking ones we think might still be valuable, then add in some other words like company names.

```
# Unique words in top 15 list
top_15_word_list <- top_words_by_ticker %>%
  pull(word) %>%
  unique() %>%
  sort()

# Vectors of both words to keep and additional ones to remove.
keep_words <- c("carbon", "covid", "emissions", "future", "government", "risk", "risk", "investmer

company_names <- c("apple", "microsoft", "visa", "united", "health", "walmart", "proctor", "gambl

# Remove the keep_words from top_15_word_list
final_word_list <- setdiff(top_15_word_list, keep_words)

# add additional words to stop list
extended_stop_words <- c(final_word_list, company_names)

extended_stop_words
```

```
 [1] "19"           "3m"           "8203"         "ability"
 [5] "actions"      "activities"   "addition"     "adverse"
 [9] "adversely"    "affect"       "aircraft"     "associates"
[13] "bottling"     "business"     "businesses"   "card"
[17] "cards"        "care"         "cash"         "cat"
[21] "chase"        "chevron"      "claims"       "class"
[25] "clients"      "clinical"     "cloud"        "commercial"
[29] "company"      "companys"     "condition"    "consumer"
[33] "content"      "continue"     "contracts"    "costs"
[37] "credit"       "customer"     "customers"    "data"
[41] "demand"       "digital"      "dtc"          "economic"
[45] "effect"       "financial"    "flows"        "food"
[49] "franchisees"  "ghg"          "health"       "healthcare"
[53] "ibm"          "impact"       "including"    "increase"
```

```
 [57] "information"   "insurance"     "ip"            "jpmorgan"
 [61] "laws"         "liquidity"     "loss"          "manufacturing"
 [65] "market"       "materially"    "materials"     "negative"
 [69] "negatively"   "oil"           "operating"     "operational"
 [73] "operations"   "partners"      "payment"       "payments"
 [77] "performance"  "pfas"          "position"      "product"
 [81] "production"   "products"      "provide"       "rates"
 [85] "regulations"  "regulatory"    "related"       "result"
 [89] "results"      "revenue"       "rights"        "risks"
 [93] "sales"        "security"      "separation"    "service"
 [97] "services"     "significant"   "software"      "stock"
[101] "subject"      "subsidiaries"  "supply"        "systems"
[105] "table"        "time"          "transactions"  "trials"
[109] "verizon"      "visa"          "apple"         "microsoft"
[113] "visa"         "united"        "health"        "walmart"
[117] "proctor"      "gamble"        "johnson"       "depot"
[121] "home"         "merck"         "coca"          "cola"
[125] "cisco"        "intel"         "disney"        "nike"
[129] "amgen"        "machines"      "salesforce"    "mcdonalds"
[133] "boeing"       "caterpillar"   "honeywell"     "american"
[137] "express"      "goldman"       "sachs"         "travelers"
[141] "dow"          "walgreens"
```

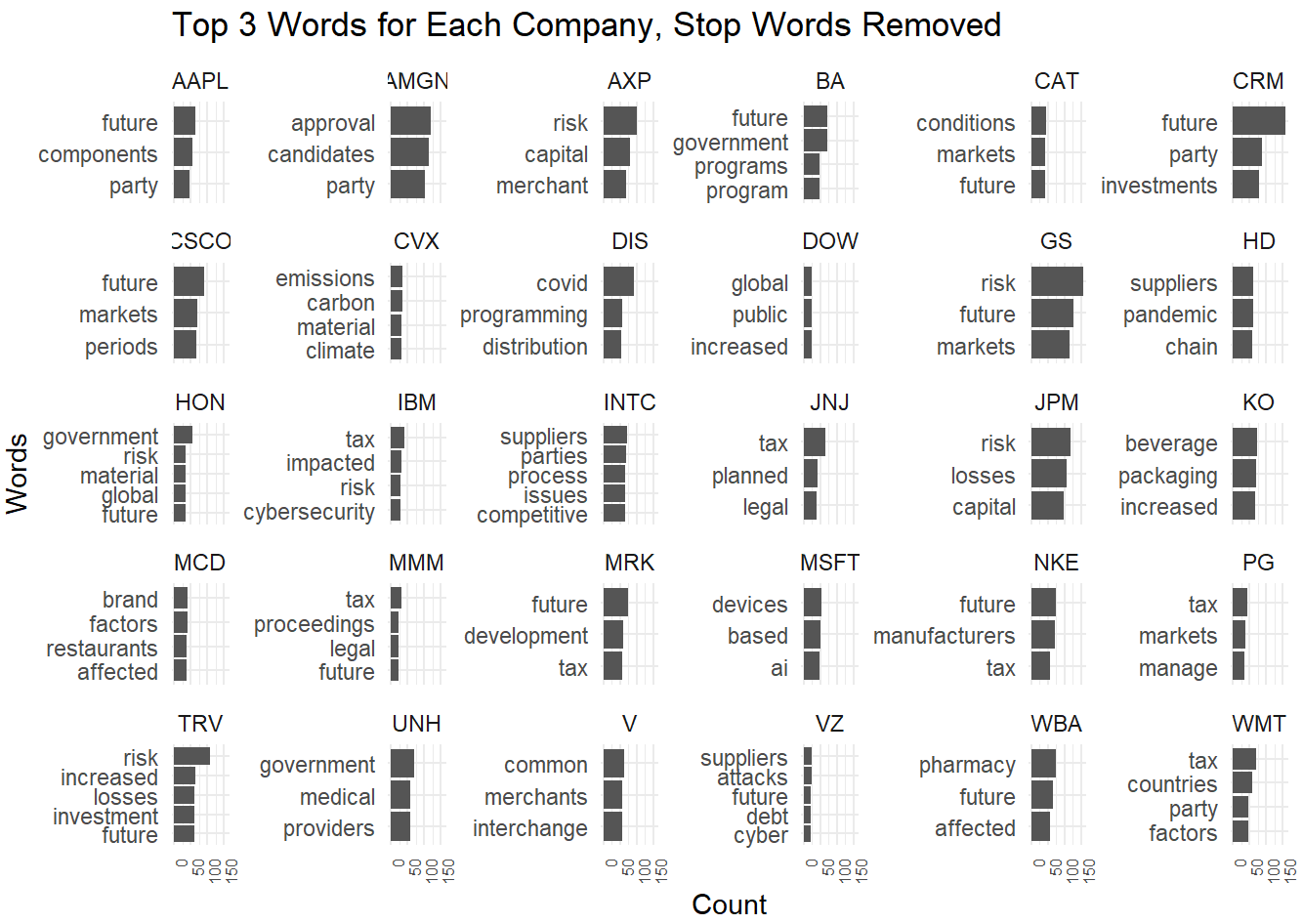
Now lets filter out rows from our tokenized data containing these Stop Words in their word column.

```
dow_risk_tokens_cleaned <- dow_risk_tokens %>%
  filter(!(word %in% extended_stop_words))
```

Nice! we're down from about 330,000 rows to 240,000. Let's rerun our faceted top 3 word chart.

Much more concise lists. What's on your mind Bill Gates?

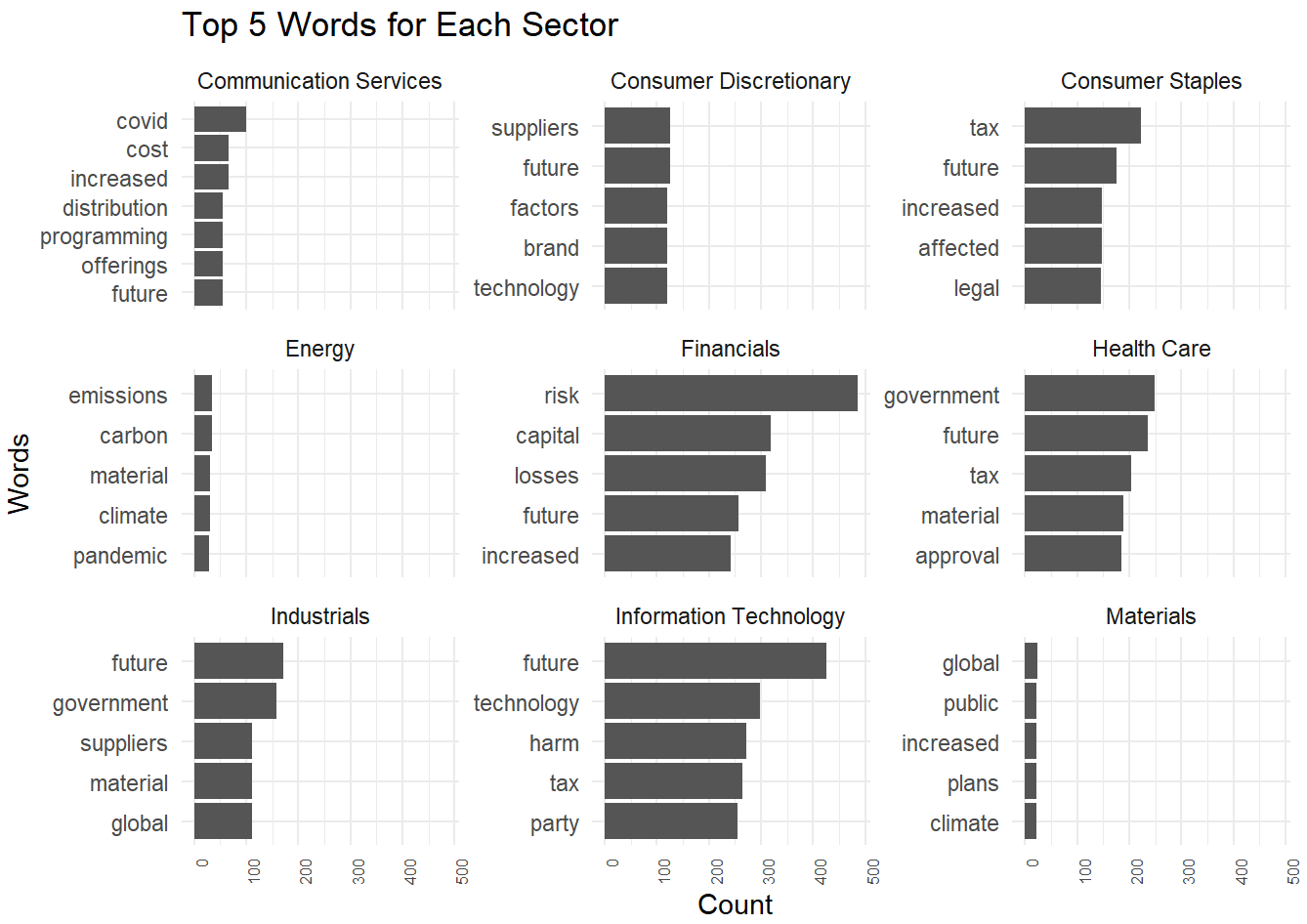
```
top_words_by_ticker_cleaned <- dow_risk_tokens_cleaned %>%
  group_by(Tickers) %>%
  count(word) %>%
  top_n(3, n) %>%
  ungroup() %>%
  arrange(Tickers, desc(n))


ggplot(top_words_by_ticker_cleaned, aes(x = n, y = reorder_within(word, n, Tickers))) +
  geom_col() +
  facet_wrap(~Tickers, scales = "free_y") +
  scale_y_reordered() +  # removes the suffixes
  labs(title = "Top 3 Words for Each Company, Stop Words Removed",
       x = "Count",
       y = "Words") +
  theme_minimal() +

  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 6))
```



Out of curiosity, let's do the same thing, but for sectors instead.

The results are intriguing as short lists of major themes in industries. Let's do the same thing again for New vs. Older 10-Ks. Take in mind some sectors have more companies than others in a limit set of 30 companies.

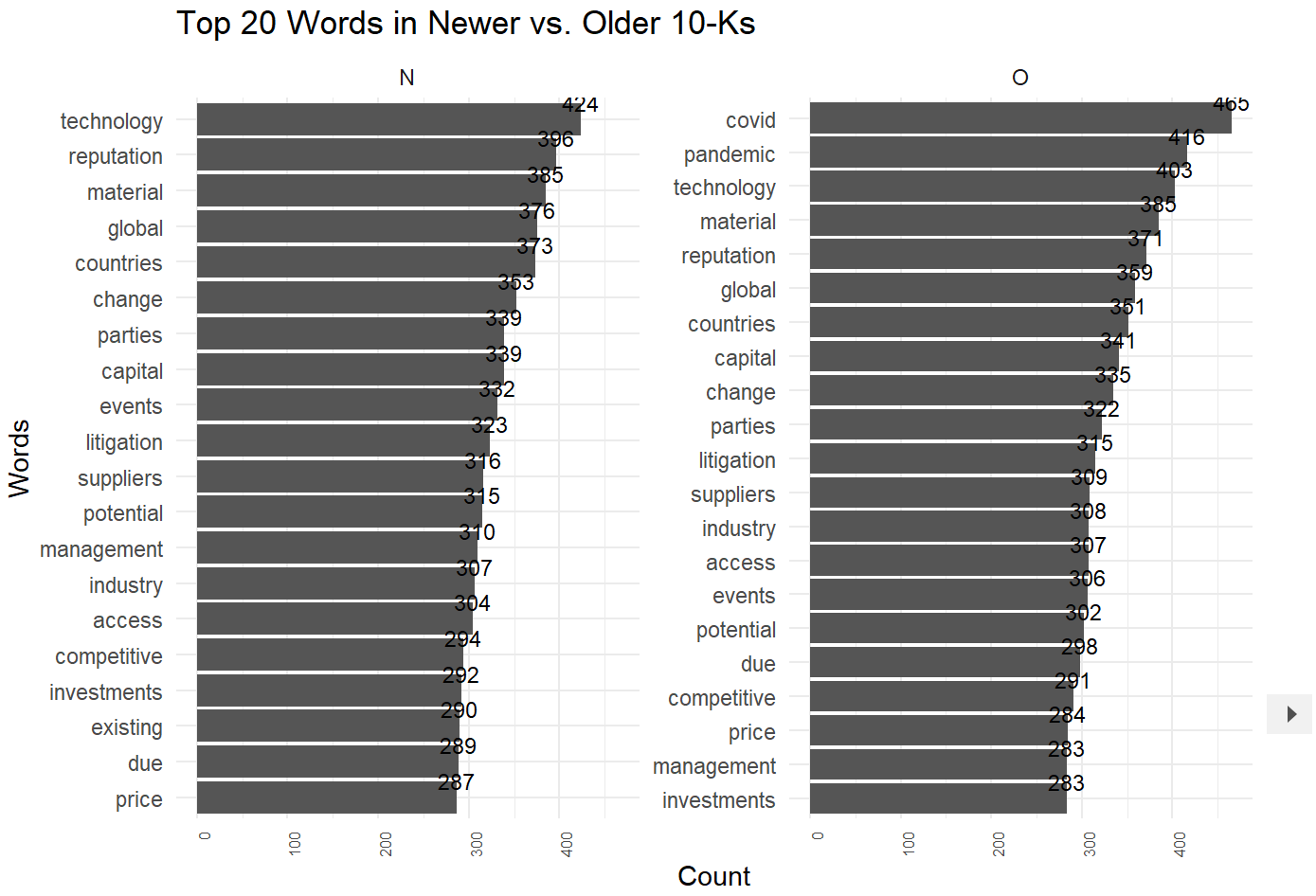
```
top_words_by_sector_cleaned <- dow_risk_tokens_cleaned %>%
  group_by(Sector) %>%
  count(word) %>%
  top_n(5, n) %>%
  ungroup() %>%
  arrange(Sector, desc(n))

ggplot(top_words_by_sector_cleaned, aes(x = n, y = reorder_within(word, n, Sector))) +
  geom_col() +
  facet_wrap(~Sector, scales = "free_y") +
  scale_y_reordered() +
  labs(title = "Top 5 Words for Each Sector",
       x = "Count",
       y = "Words") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 6))
```



We're running out of steam in how interesting these faceted charts are, but clearly COVID and the Pandemic were of top concern two 10-Ks ago but have fallen away in recent ones, leaving behind only concern about the supply of production inputs.

```
# Our first go yieled boilerplate lanugage recycled year after year, so we're going to remove some

more_stop_words = c("future","risk","increased","tax","markets","affected","conditions", "governme

dow_risk_tokens_cleaned_again <- dow_risk_tokens_cleaned %>%
  filter(!(word %in% more_stop_words))

top_words_by_period_cleaned <- dow_risk_tokens_cleaned_again %>%
  group_by(New_or_Old) %>%
  count(word) %>%
  top_n(20, n) %>%
  ungroup() %>%
  arrange(New_or_Old, desc(n))

ggplot(top_words_by_period_cleaned, aes(x = n, y = reorder_within(word, n, New_or_Old))) +
  geom_col() +
  geom_text(aes(label = n), position = position_dodge(width = 0.9), vjust = -0.5, size = 3) +
  facet_wrap(~New_or_Old, scales = "free_y") +
  scale_y_reordered() +  # This function removes the suffixes
  labs(title = "Top 20 Words in Newer vs. Older 10-Ks",
```

```
      x = "Count",
      y = "Words") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 6))
```



Did the sections increase in length from older to newer 10-Ks?

'N' represents the newer 10-Ks while 'O' represents ones from the year prior. The average risk factor word count over these time periods increased on average by about 137 words or 1.28%. Do these companies generally have more risks, are risk disclosures just growing in length over time, or is this simply an insignificant difference?

```
library(dplyr)

dow_risk_factors_cleaned %>%
  mutate(word_count = str_count(text, "\\S+")) %>%
  group_by(New_or_Old) %>%
  summarise(average_word_count = mean(word_count))
```

```
# A tibble: 2 × 2
  New_or_Old average_word_count
  <chr>                   <dbl>
```

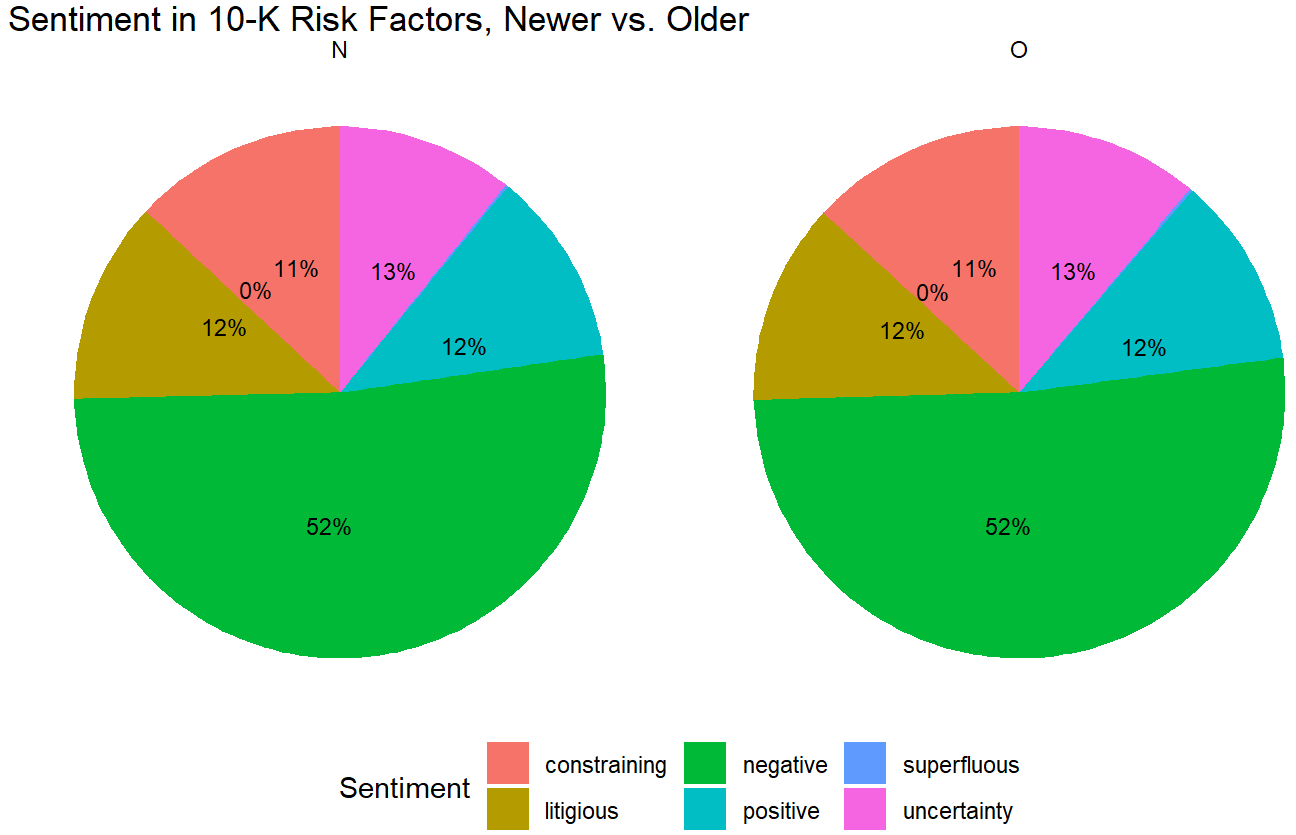
```
1 N                     10830.
2 O                     10692.
```

How do sentiments differ by: older to newer documents, sector, company?

As it was developed with financial documents specifically in mind, we will first use 'Loughran' sentiment lexicon as our core tool for measuring sentiment in these risk factor sections.

The results between older and newer documents are almost identical, with percents differing by the tenth and hundredth decimal places. It's not surprising as 10-Ks are often copy and paste jobs. After inner-joining, our number of analyze-able rows decreases to ~40,000 containing only bland business vocabulary, which also limits the potential for sentiment diversity.

```
# Remember this tibble has both batches of our stop words removed, let's also remove numbers just
dow_risk_tokens_cleaned_again_filtered <- dow_risk_tokens_cleaned_again %>%
  filter(!grepl("\\d", word))

# inner join sentiments and complete percentages of the group whole
sentiment_data <- dow_risk_tokens_cleaned_again_filtered %>%
  inner_join(get_sentiments("loughran"),relationship = "many-to-many") %>%
  count(New_or_Old, sentiment) %>%
  group_by(New_or_Old) %>%
  mutate(percent = n / sum(n),
         label_position = cumsum(percent) - 0.5 * percent)

# Create faceted pie charts with labels
ggplot(sentiment_data, aes(x = "", y = percent, fill = sentiment)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y") +
  geom_text(aes(y = label_position, label = scales::percent(percent, accuracy = 1)), size = 3) +
  facet_wrap(~New_or_Old) +
  labs(title = "Sentiment in 10-K Risk Factors, Newer vs. Older",
       x = NULL,
       y = NULL,
       fill = "Sentiment") +
  theme_void() +
  theme(legend.position = "bottom")
```



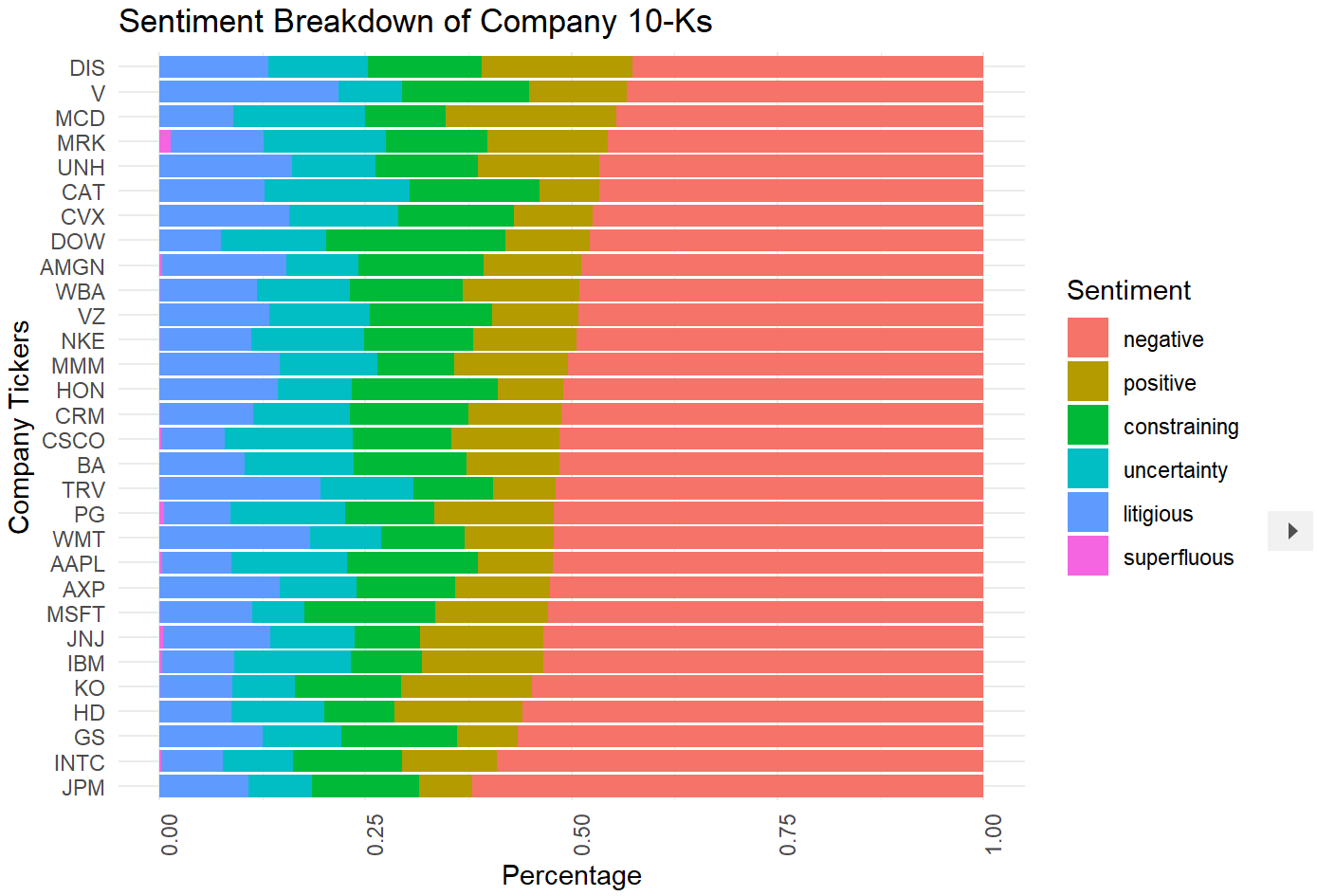
Let's do the same for companies and sector.

Company

Disney is unsurprisingly the least negative. McDonalds is the most positive. JP Morgan is the most negative and the least positive. DOW is the most constraining, Johnson and Johnson is the least. Caterpillar is the most uncertain, Microsoft is the least. Visa is the most litigious, DOW is the least. Merck has been feeling a bit superfluous.

```
loughran_sentiment_data_tickers <- dow_risk_tokens_cleaned_again_filtered %>%
  inner_join(get_sentiments("loughran"), relationship = "many-to-many") %>%
  count(Tickers, sentiment) %>%
  group_by(Tickers) %>%
  mutate(percent = n / sum(n))

# Define the order of sentiments
sentiment_order <- c("negative", "positive", "constraining", "uncertainty", "litigious", "superflu

# Calculate the total negative sentiment for each Ticker
negative_sentiment_totals <- loughran_sentiment_data_tickers %>%
  filter(sentiment == "negative") %>%
  ungroup() %>%
  arrange(desc(percent)) %>%
  mutate(Tickers = factor(Tickers, levels = Tickers))
```

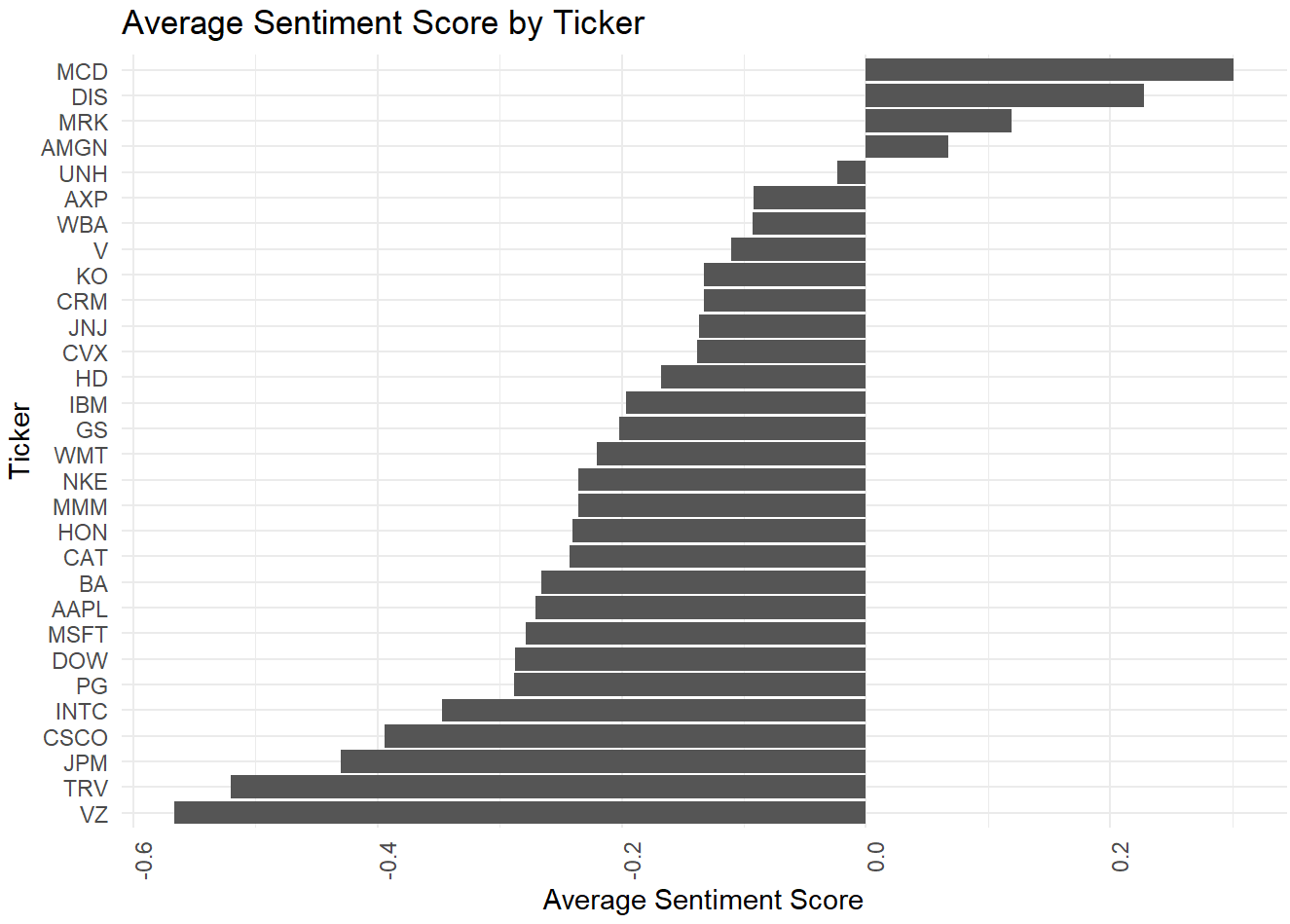
```r
# Set the levels of sentiment factor
loughran_sentiment_data_tickers$sentiment <- factor(loughran_sentiment_data_tickers$sentiment, le

# Create stacked bar chart ordered by negative sentiment with value labels
ggplot(loughran_sentiment_data_tickers, aes(x = Tickers, y = percent, fill = sentiment)) +
  geom_bar(stat = "identity", position = "stack") +
  coord_flip() +
  scale_x_discrete(limits = negative_sentiment_totals$Tickers) +
  labs(title = "Sentiment Breakdown of Company 10-Ks",
       x = "Company Tickers",
       y = "Percentage",
       fill = "Sentiment") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



The Afinn lexicon only yields ~23,000 rows to analyze, but let's take a look at average Afinn score by company anyways.

Disney and McDonald's have firmly planted their flags in Dow-world as *happy* companies. Verizon sticks out as having dropped to dead last. Some shuffling occurred in the middle of the pack, but nothing too significant.

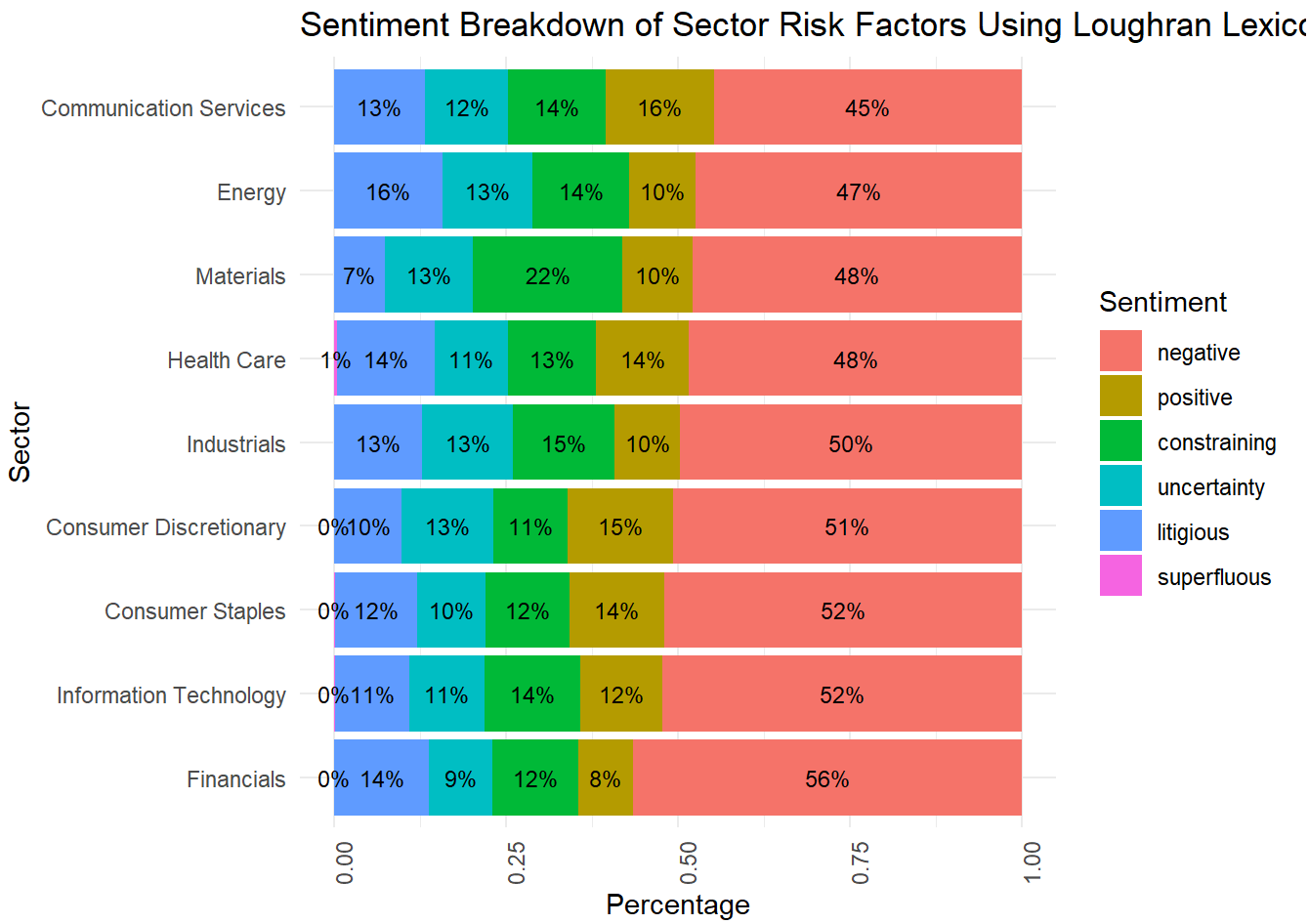
```r
# Assuming dow_risk_tokens_cleaned_again_filtered is already created
# Join with AFINN lexicon
afinn_sentiment_data <- dow_risk_tokens_cleaned_again_filtered %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  group_by(Tickers) %>%
  summarise(average_sentiment = mean(value)) %>%
  ungroup() %>%
  arrange(desc(average_sentiment))  # Sort by average sentiment in descending order

# Plot the average sentiment score for each ticker
ggplot(afinn_sentiment_data, aes(x = reorder(Tickers, average_sentiment), y = average_sentiment)) +
  geom_col() +
  coord_flip() +  # Flip coordinates for horizontal layout
  labs(title = "Average Sentiment Score by Ticker",
       x = "Ticker",
       y = "Average Sentiment Score") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Sectors

These results echo those of the company results. Financials interestingly enough have the highest negative sentiment, but also the lowest percent of uncertainty. What does Jamie Dimon know that we don't?

```r
loughran_sentiment_data_sectors <- dow_risk_tokens_cleaned_again_filtered %>%
  inner_join(get_sentiments("loughran"),relationship = "many-to-many") %>%
  count(Sector, sentiment) %>%
  group_by(Sector) %>%
  mutate(percent = n / sum(n))

# order of sentiments on chart
sentiment_order <- c("negative", "positive", "constraining", "uncertainty", "litigious", "superflu

# Calculate the total negative sentiment for each Ticker
negative_sentiment_totals <- loughran_sentiment_data_sectors %>%
  filter(sentiment == "negative") %>%
  ungroup() %>%
  arrange(desc(percent)) %>%
  mutate(Sector = factor(Sector, levels = Sector))


loughran_sentiment_data_sectors$sentiment <- factor(loughran_sentiment_data_sectors$sentiment, lev

# Create stacked bar chart ordered by negative sentiment with value labels
ggplot(loughran_sentiment_data_sectors, aes(x = Sector, y = percent, fill = sentiment)) +
  geom_bar(stat = "identity", position = "stack") +
  geom_text(aes(label = scales::percent(percent, accuracy = 1)),
            position = position_stack(vjust = 0.5), size = 3) +
  coord_flip() +
  scale_x_discrete(limits = negative_sentiment_totals$Sector) +
  labs(title = "Sentiment Breakdown of Sector Risk Factors Using Loughran Lexicon",
       x = "Sector",
       y = "Percentage",
       fill = "Sentiment") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

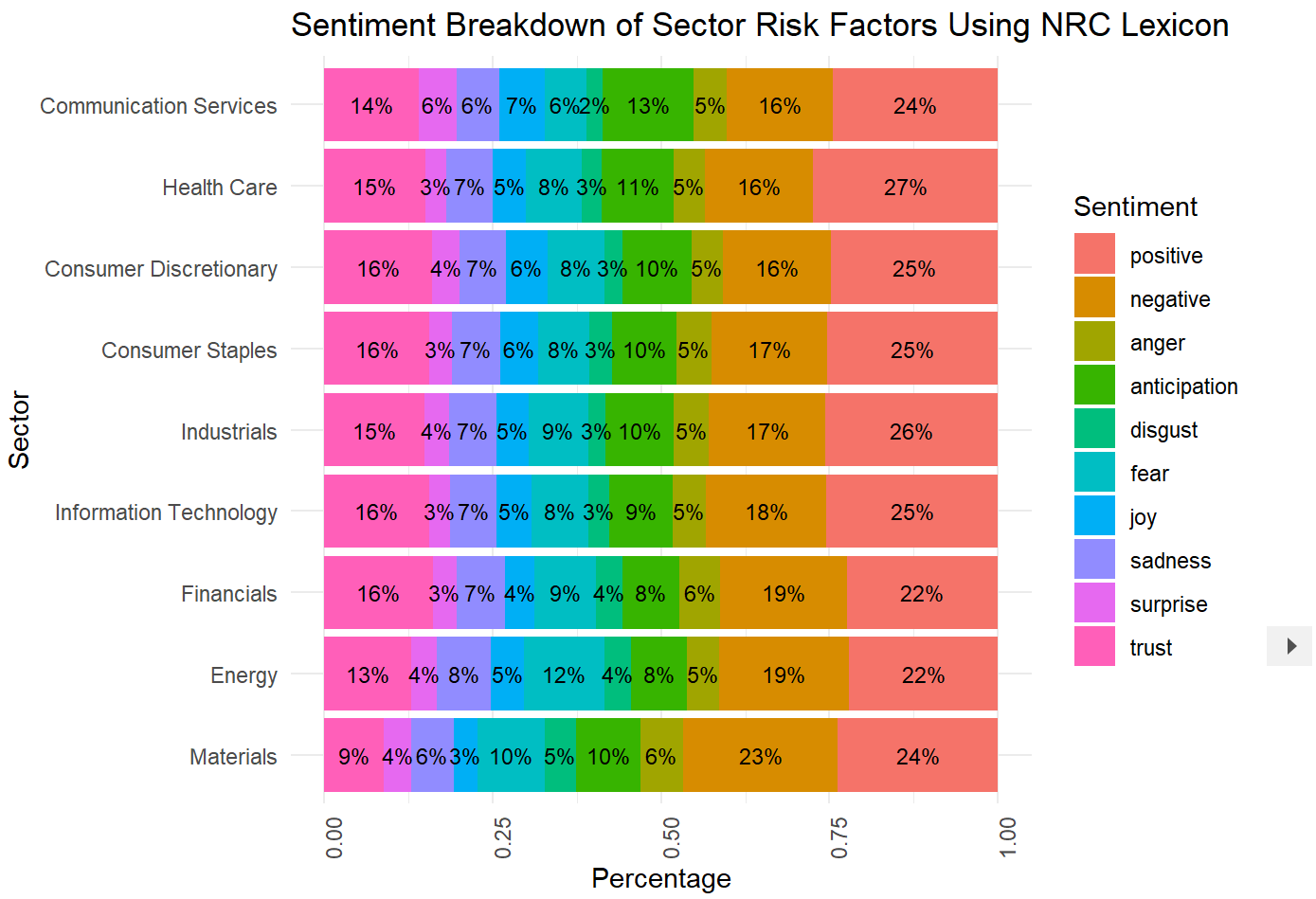


Let's look at sectors but using the 'NRC' lexicon instead, which leaves ~100,000 rows, to see if we obtain synonymous results.

There appears to be less variance in sector sentiments using the NRC lexicon. We do have a greater breadth of sentiments to explore.

```
nrc_sentiment_data_sectors <- dow_risk_tokens_cleaned_again_filtered %>%
  inner_join(get_sentiments("nrc"), relationship = "many-to-many") %>%
  count(Sector, sentiment) %>%
  group_by(Sector) %>%
  mutate(percent = n / sum(n))

sentiment_order <- c("positive", "negative", "anger", "anticipation", "disgust", "fear", "joy", "

# Calculate the total negative sentiment for each Sector
negative_sentiment_totals <- nrc_sentiment_data_sectors %>%
  filter(sentiment == "negative") %>%
  ungroup() %>%
  arrange(desc(percent)) %>%
  mutate(Sector = factor(Sector, levels = Sector))

# Set the levels of sentiment factor
nrc_sentiment_data_sectors$sentiment <- factor(nrc_sentiment_data_sectors$sentiment, levels = sen
```

```r
# stacked bar chart
ggplot(nrc_sentiment_data_sectors, aes(x = Sector, y = percent, fill = sentiment)) +
  geom_bar(stat = "identity", position = "stack") +
  geom_text(aes(label = scales::percent(percent, accuracy = 1)),
            position = position_stack(vjust = 0.5), size = 3) +
  coord_flip() +
  scale_x_discrete(limits = negative_sentiment_totals$Sector) +
  labs(title = "Sentiment Breakdown of Sector Risk Factors Using NRC Lexicon",
       x = "Sector",
       y = "Percentage",
       fill = "Sentiment") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

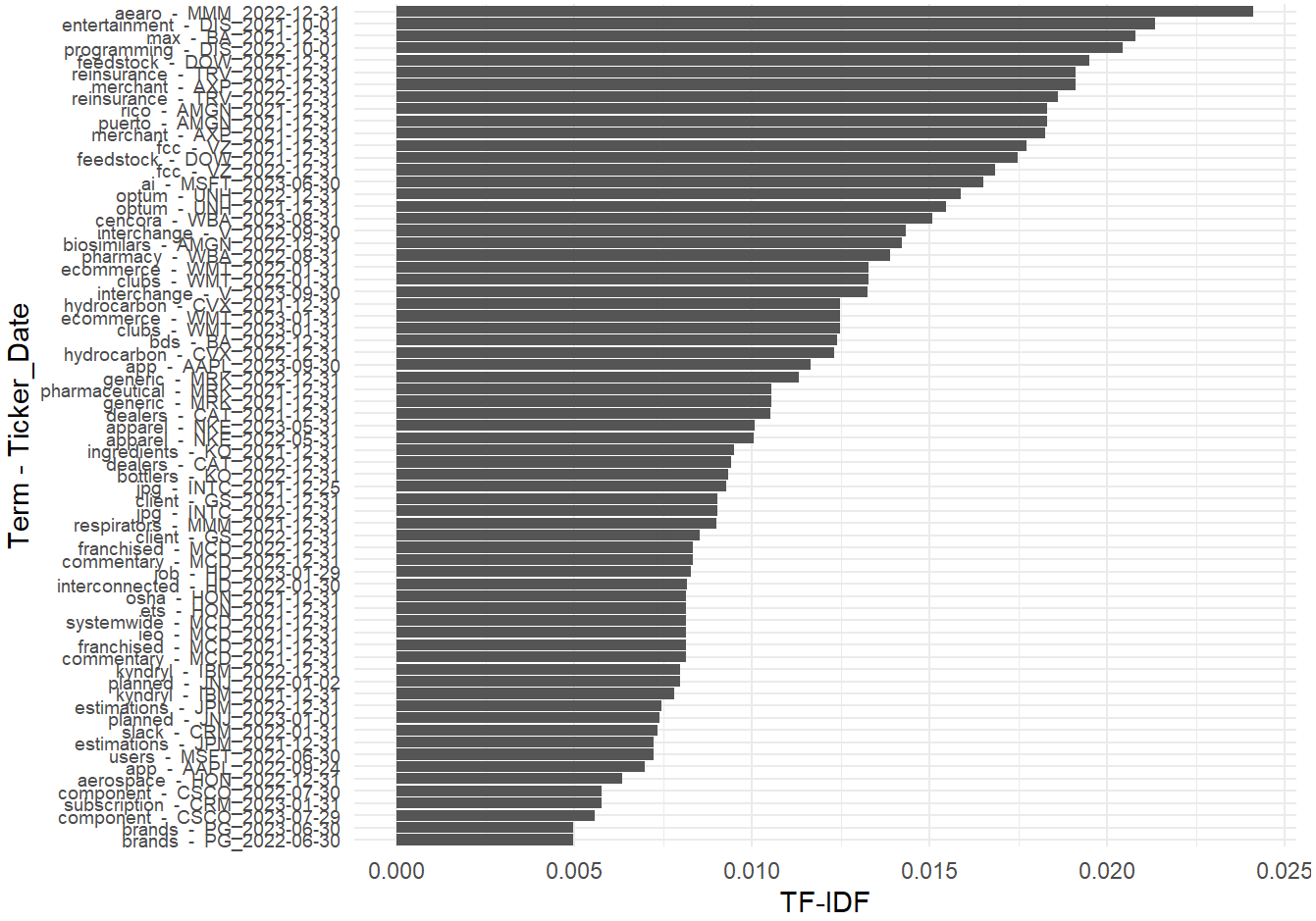


What unique risks are these companies facing?

Through TF-IDF, we can augar our way down to the real uglies buried under corporate cliches.

3M sites risks related to a ear protection lawsuit. Boeing is pretty sure it's 737 is good to go. Amgen is doing something in Puerto Rico. Walgreens recently sold off a large stake in Cencora. Walmart feels Costco is on to something. This list is littered with acquisitions, spinoffs, ventures, and insecurities.

```r
# add in a few more stop words
tf_idf_stop_words <- c("ibm","chevrons", "restaurants","restaurant", "ot", "intc","wireless","bev

dow_risk_tokens_cleaned_again_filtered %>%
  filter(!(word %in% tf_idf_stop_words)) %>%
  count(ticker_date, word) %>%
  bind_tf_idf(term = word, document = ticker_date, n = n) %>%
  group_by(ticker_date) %>%
  slice_max(order_by = tf_idf, n = 1) %>%
  mutate(combined_term = paste(word, " - ", ticker_date)) %>%
  ggplot(aes(x = tf_idf, y = reorder(combined_term, tf_idf))) +
  geom_col() +
  labs(x = "TF-IDF", y = "Term - Ticker_Date") +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 7))
```



What are common risks shared by companies?

Trigrams for Thematic Comprehension

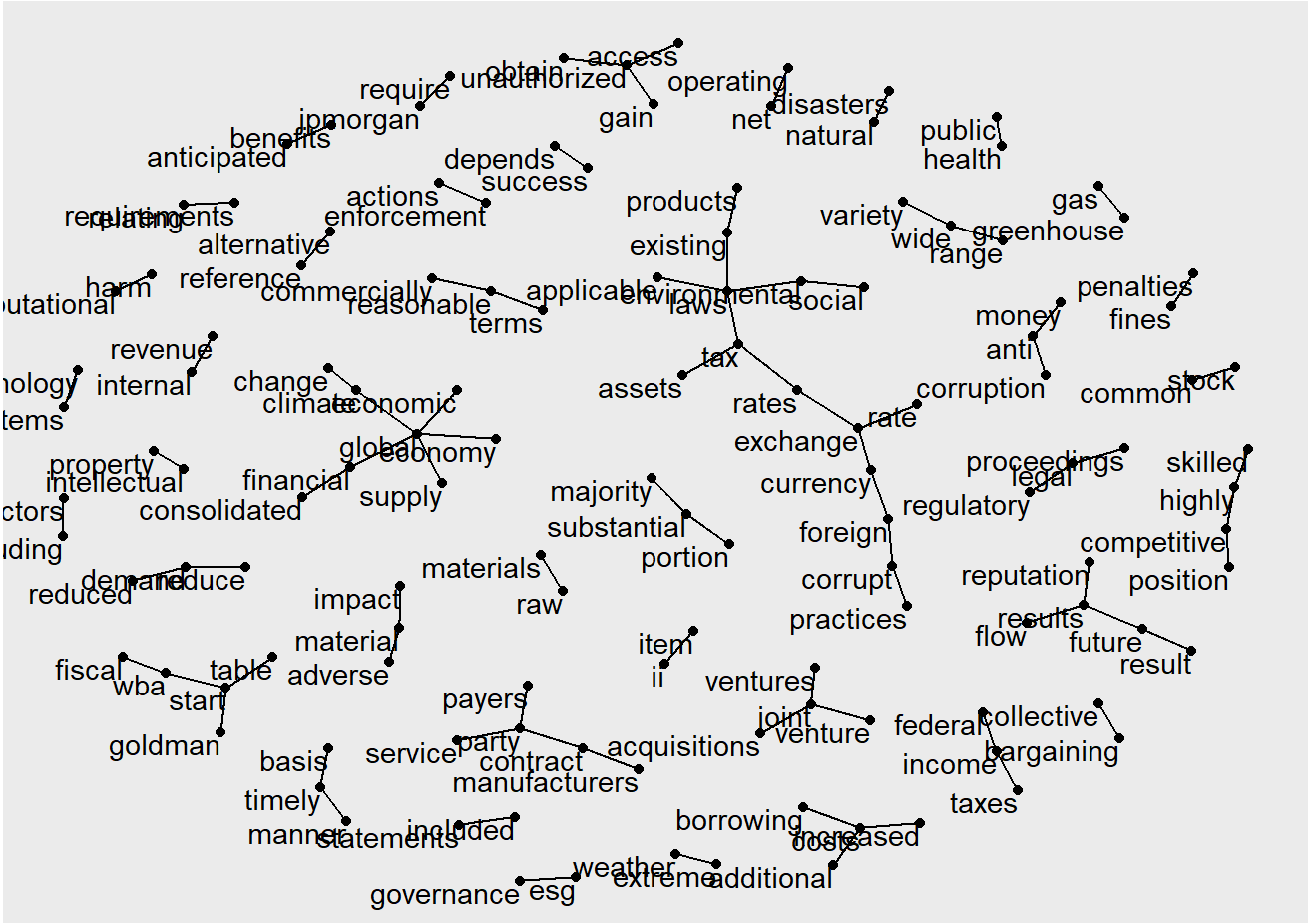
Bigrams results were lacking dimension, so we used trigrams instead. We see some distinct thematic constellations that brightly illustrate CEO night terrors:

- Contracts and payment terms
- Protection of intellectual property
- Finding enough skilled labor
- Tax burdens
- Financial reputation & consistency
- Reduction in product demand
- Success of new ventures
- Increased borrowing costs
- Security breaches
- Effect of climate change on global economy
- Foreign corruption and its effects on exchange rates and trade performance
- Collective bargaining
- Legal woes

And more!

```
library(ggraph)
library(igraph)

# Form trigrams and remove stop words and numbers
tigram_count <- dow_risk_factors_cleaned %>%
  unnest_tokens(output = trigrams,
                input = text,
                token = "ngrams",
                n = 3) %>%
  count(trigrams,sort = T) %>%
   separate(col = trigrams,
            into = c("word1","word2","word3"),
            sep = " ") %>%
  filter(!is.na(word1)) %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(!word1 %in% extended_stop_words) %>%
  filter(!word2 %in% more_stop_words) %>%
  filter(!grepl("\\d", word1)) %>%
  filter(!grepl("\\d", word2))
```

```
tigram_count %>%
  filter(n > 18) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = name),
                 vjust = 1,
                 hjust = 1)
```



Topic Modeling

Load libraries and construct our document term matrix.

```
library(ldatuning)
library(topicmodels)
```

```
dtm <- dow_risk_tokens_cleaned_again_filtered %>%
  count(ticker_date, word) %>%
  cast_dtm(ticker_date, word, n)
```

```
dtm
```

```
<<DocumentTermMatrix (documents: 60, terms: 7497)>>
Non-/sparse entries: 77405/372415
Sparsity           : 83%
Maximal term length: 21
Weighting          : term frequency (tf)
```

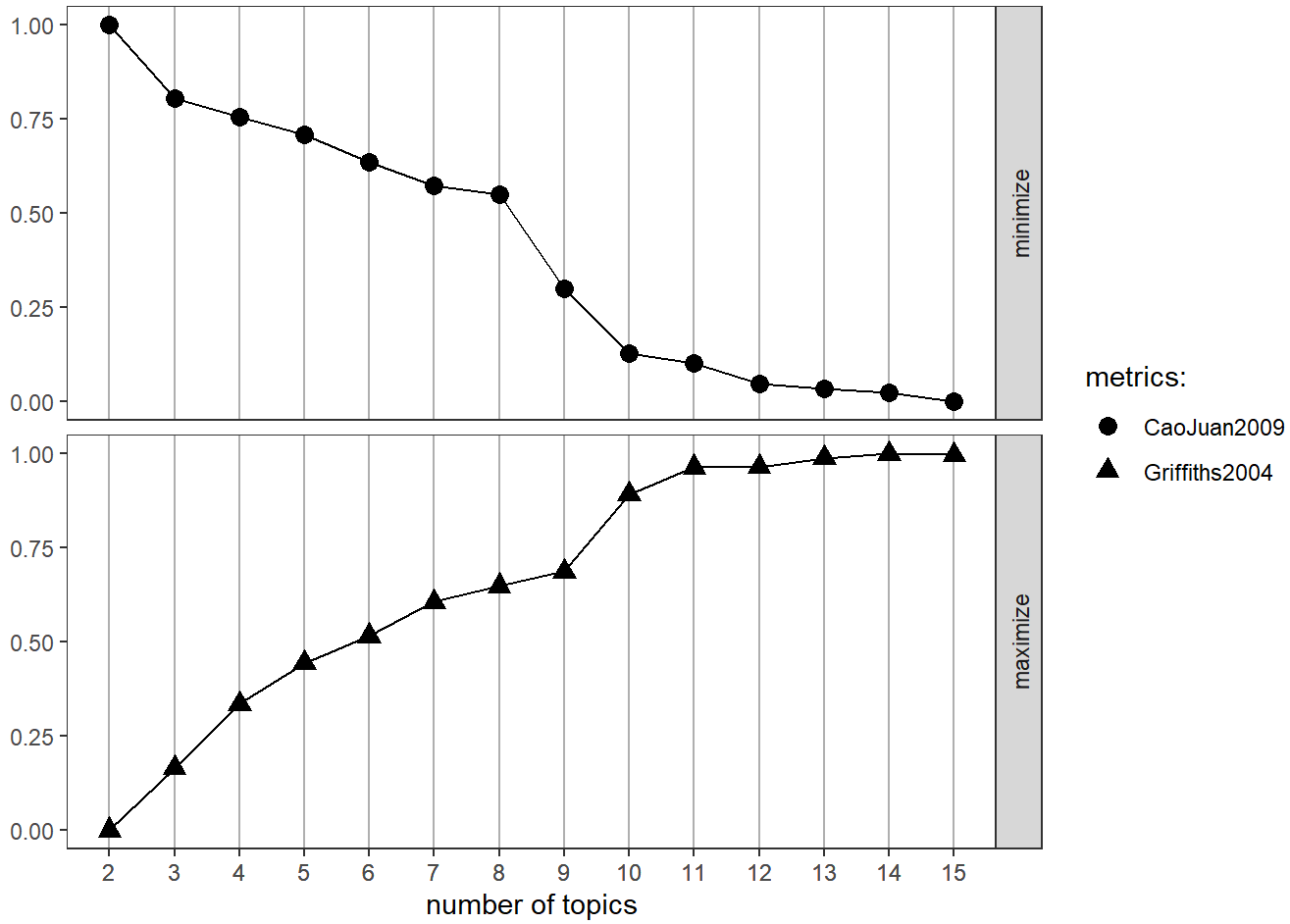
Here we use the 'FindTopicsNumber' function in 'ldatuning' to determine what might be the 'optimal' number of topics. CaoJuan2009, a metric for measuring topic performance indicates there is not much more improvement after 8 topics.

However, after having modeled 8 topics, its was clear there was too much qualitative overlap between many groups, so we will select 4 topics instead. This is a good example of art trumping science in data analysis. Or, maybe humans just lack the RAM to see differences that are obvious to a computer.

```
topics <- seq(from = 2, to = 15, by = 1)

# Calculate metrics for each number of topics
fit_models <- ldatuning::FindTopicsNumber(
  dtm,
  topics = topics,
  metrics = c("CaoJuan2009",  "Griffiths2004"),,
  method = "Gibbs",
  control = list(seed = 77),
  mc.cores = 2, # Adjust this according to your machine's capabilities
  verbose = TRUE
)
```

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

```
ldatuning::FindTopicsNumber_plot(fit_models)
```



We've already made our dtm (document term matrix) above, now we are going to apply LDA.

```
risk_factors_lda <- LDA(dtm,
                        k = 4,
                        control = list(seed = 1234))

risk_factors_lda
```

```
A LDA_VEM topic model with 4 topics.
```

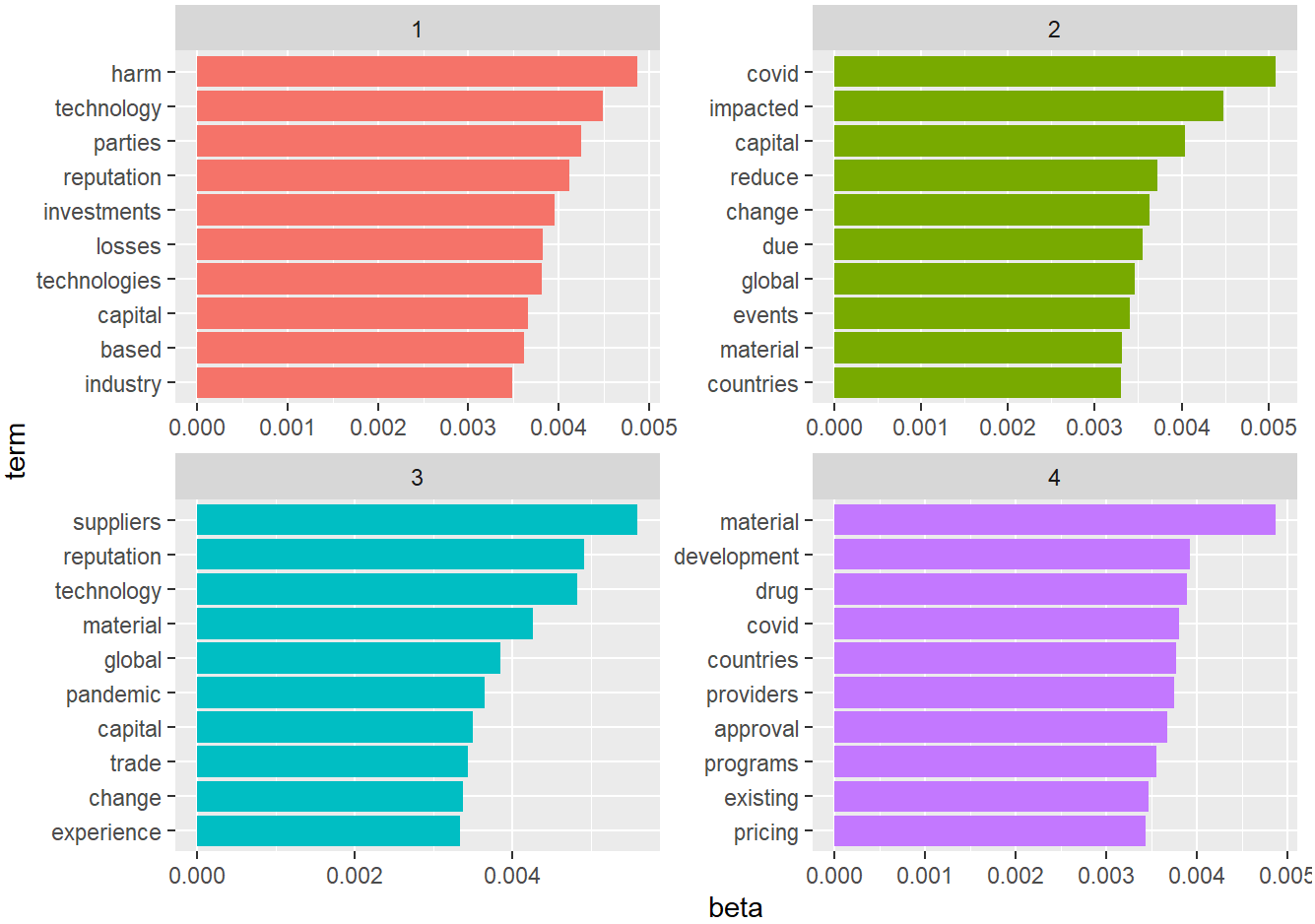
We will now model with a beta parameter, to understand words within topics better. Look at the top 10 words for the 4 topics our model has created.

It's safe to say topic modeling with beta parameters in a body of such similar documents may have a difficult time presenting notion-shattering insights.

Topic 1 seems focused on technology and losses surrounding it. 2 can be the COVID topic. 3 and 4 are more focused on global events and supply chains.

```
risk_factor_topics <- tidy(risk_factors_lda,
                           matrix = "beta")

top_terms <- risk_factor_topics %>%
  group_by(topic) %>%
```

```
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)

top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(x = beta, y = term, fill = as.factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic , scales = "free" ) +
  scale_y_reordered()
```

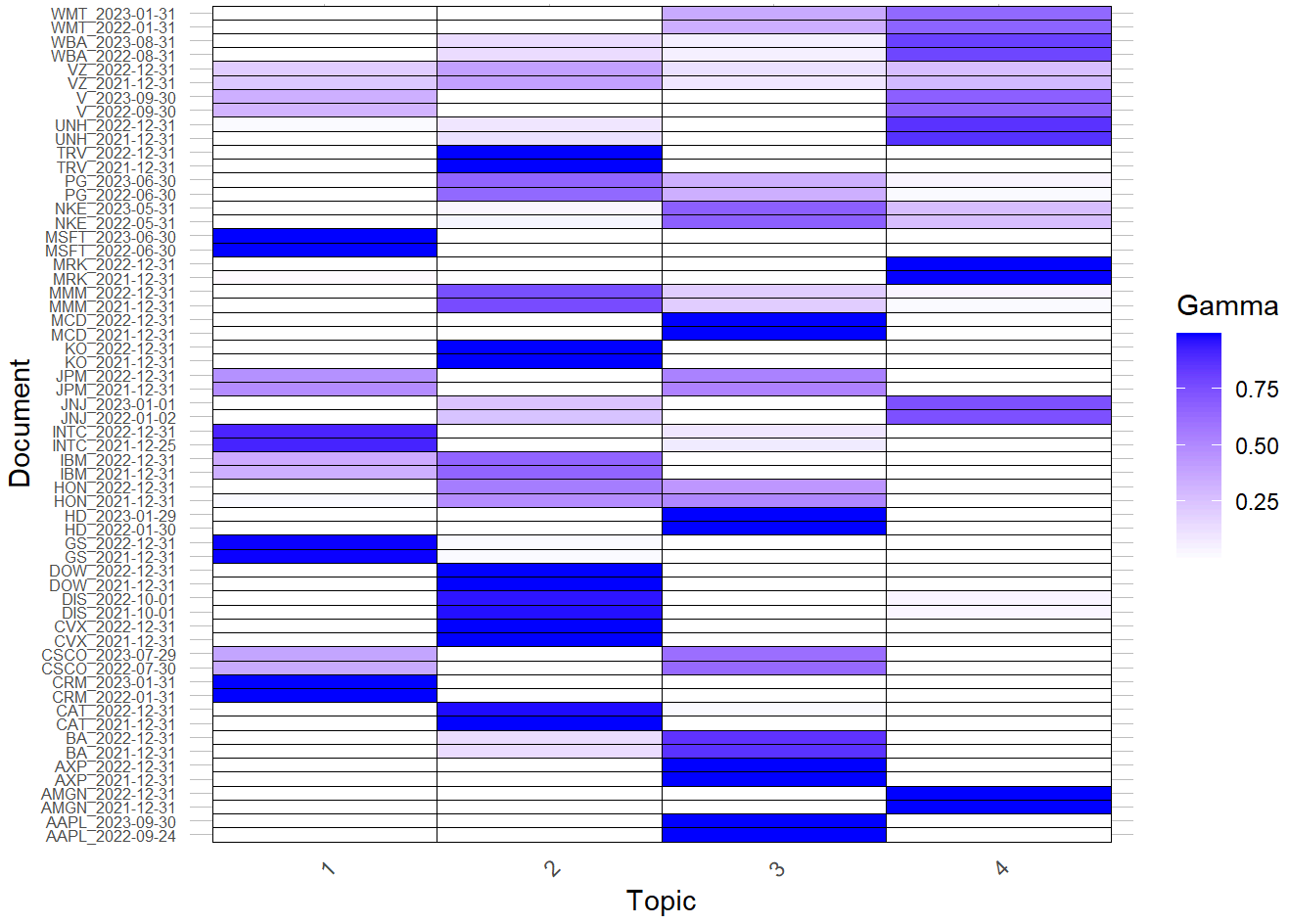


We can also run a model with gamma parameters to better-understand the distribution of topics in the corpus of documents. With so many documents, it can be difficult to see on a single chart. Though when closely examined, this heatmap is powerful in that it tells us the difference from company to company and from document to document.

For large changes in 10-K Risk Factor topics, we're looking for stark contrast in coloring from one 10-K to the next for the same company, and we don't see much difference at all for any company.

```
#Get document-topic probabilities
risk_factors_gamma <- tidy(risk_factors_lda,
                           matrix = "gamma")
# pivot_wider
wide_data <- risk_factors_gamma %>%
```

```
  pivot_wider(id_cols = document, names_from = topic, values_from = gamma)

long_data_for_plot <- wide_data %>%
  pivot_longer(-document, names_to = "topic", values_to = "gamma")

# Create the heatmap with outlined boxes
ggplot(long_data_for_plot, aes(x = topic, y = document, fill = gamma)) +
  geom_tile(colour = "black") +  # Add a black outline to each tile
  scale_fill_gradient(low = "white", high = "blue") +
  labs(x = "Topic", y = "Document", fill = "Gamma") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text.y = element_text(size = 6),
        panel.grid.major = element_line(color = "grey", size = 0.1),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())
```



Which companies risk factors changed the least and most year over year?

Our goal here is to calculate the difference from one document to the next. To do this, we bring our dtm into a matrix and perform cosine similarity calculations for each possible vector pairing. We are left with a 60x60 matrix we then want to convert to a human-readable table.

What we've found is that Nike, Travelers, and Proctor and Gamble changed the least with near 99% cosine similarity, while Boeing, Honeywell, and 3M changed the most, with 3M being a complete outlier at only 79% similar. I guess we'll actually have to read their 10-Ks now.

What we then see is companies in similar industries and sectors having relatively high similarity measures around 70-75%. Goldman Sachs and JP Morgan for example.

At the bottom are similarity measures around 30% with pairings of very dissimilar companies. DOW Chemical pairings make up a fairly large percent of the bottommost pairings, indicating there must be some pretty unique verbiage in 10-K risk factors for the producers of DDT.

```
library(proxy)
library(tm)
library(slam)

# Similarity calculations
dtm_matrix <- as.matrix(dtm)
dtm_norm <- dtm_matrix / sqrt(rowSums(dtm_matrix^2))
similarity_matrix <- proxy::simil(dtm_norm, method = "cosine")

# convert to matrix
similarity_full_matrix <- as.matrix(similarity_matrix)

# document names
document_names <- attr(similarity_matrix, "Labels")

# Assign row and column names to the matrix
rownames(similarity_full_matrix) <- document_names
colnames(similarity_full_matrix) <- document_names

# Create an empty dataframe
similarity_pairs <- data.frame(document1 = character(),
                               document2 = character(),
                               similarity = numeric(),
                               stringsAsFactors = FALSE)

# Loop through the upper triangular part of the matrix
for (i in 1:(nrow(similarity_full_matrix) - 1)) {
  for (j in (i + 1):ncol(similarity_full_matrix)) {
    similarity_pairs <- rbind(similarity_pairs,
                              data.frame(document1 = document_names[i],
                                         document2 = document_names[j],
                                         similarity = similarity_full_matrix[i, j]))
  }
}

# create two sorted data frames
similarity_list <- similarity_pairs %>%
  arrange(similarity)
```

```r
similarity_list_desc <- similarity_pairs %>%
  arrange(desc(similarity))

# The most similar 10-K Risk Factors
head(similarity_list_desc, 20)
```

```
         document1       document2 similarity
1   NKE_2022-05-31  NKE_2023-05-31  0.9876396
2   TRV_2021-12-31  TRV_2022-12-31  0.9866124
3    PG_2022-06-30   PG_2023-06-30  0.9860760
4   IBM_2021-12-31  IBM_2022-12-31  0.9851925
5   JPM_2021-12-31  JPM_2022-12-31  0.9828406
6   CRM_2022-01-31  CRM_2023-01-31  0.9807219
7   UNH_2021-12-31  UNH_2022-12-31  0.9805250
8   CAT_2021-12-31  CAT_2022-12-31  0.9802595
9  CSCO_2022-07-30 CSCO_2023-07-29  0.9796366
10 AAPL_2022-09-24 AAPL_2023-09-30  0.9791733
11    V_2022-09-30    V_2023-09-30  0.9767659
12 INTC_2021-12-25 INTC_2022-12-31  0.9763322
13 AMGN_2021-12-31 AMGN_2022-12-31  0.9759854
14  AXP_2021-12-31  AXP_2022-12-31  0.9740782
15   VZ_2021-12-31   VZ_2022-12-31  0.9722402
16   GS_2021-12-31   GS_2022-12-31  0.9722202
17  MRK_2021-12-31  MRK_2022-12-31  0.9710405
18   KO_2021-12-31   KO_2022-12-31  0.9675583
19  MCD_2021-12-31  MCD_2022-12-31  0.9667834
20   HD_2022-01-30   HD_2023-01-29  0.9643386
```

```r
# The least similar 10-K Risk Factors
head(similarity_list, 20)
```

```
         document1       document2 similarity
1   DOW_2021-12-31    V_2022-09-30  0.2998458
2   DOW_2022-12-31    V_2022-09-30  0.3050195
3   DIS_2021-10-01  DOW_2022-12-31  0.3098876
4   DOW_2021-12-31    V_2023-09-30  0.3115431
5   DOW_2022-12-31    V_2023-09-30  0.3160594
6   DOW_2021-12-31 MSFT_2023-06-30  0.3260596
7   DIS_2022-10-01  DOW_2022-12-31  0.3299174
8   DOW_2021-12-31  UNH_2022-12-31  0.3326861
9   DOW_2022-12-31 MSFT_2023-06-30  0.3335139
10 AMGN_2021-12-31  DOW_2022-12-31  0.3344341
11  DOW_2021-12-31 MSFT_2022-06-30  0.3382896
12  DOW_2022-12-31  UNH_2022-12-31  0.3384555
13  DOW_2022-12-31  UNH_2021-12-31  0.3415138
14  DOW_2022-12-31 MSFT_2022-06-30  0.3420624
15  DIS_2021-10-01  DOW_2021-12-31  0.3451689
16 AMGN_2022-12-31  DOW_2022-12-31  0.3456852
17  DIS_2021-10-01  MRK_2022-12-31  0.3459382
18  DIS_2021-10-01  UNH_2022-12-31  0.3463041
```

19	DOW_2021-12-31	UNH_2021-12-31	0.3483355
20	DIS_2021-10-01	MCD_2022-12-31	0.3484825

Conclusion

Through this analysis, we are able to make some key findings.

In our EDA, saw the jungle of corporate jargin on display, and were able to craft some useful stop word lists to correct for that. We also saw thematic sector interests.

We saw that 10-K risk factor sections got a little bit longer in the most recent batch of 10-Ks, but needed further analysis to determine why. We saw material differences in the outlooks for different industries and companies, but not much change in sentiment from one year to the next as was initially hypothesized. Companies that have branded themselves in a happy light portray that down to the 10-K. Negative sentiments shown by JP Morgan and Goldman Sachs show that interest rate risk is a very real thing.

We used TF-IDF to pinpoint unique, non-recurring vocabulary for many of these companies. This was the first time in the analysis we were able to break through the boilerplate to see various lawsuits and ventures with uncertain outcomes. I could see this method being of use to institutional analysts who cover specific companies and sectors.

On the flip side, the boilerplate nature of these disclosures came in handy when using trigrams to identify common themes among all stated risk factors. We were able to generate a comprehensive list of issues present within not one, but many of these companies. Our topic modeling was less fruitful, though the gamma distributions provided a nice visual representation of where different companies are the most concerned.

Finally we used Cosine Similarity to compute the similarity between all 60 10-K documents in this analysis. Predictably, most companies were over 90% similar from year to year, but there were some notable outliers who deserve a closer look, 3M for example.

As a legal obligation, these companies generate a lot of fluffy text that must be condensed and dissected for real insights. This analysis proved that certain models can provide a framework for making that easier. While we've made a great start, there is much more than can be done to understand the predictive power of these words.

What's for certain is that these companies are very aware of and planning for the stated macroeconomic trends.

More Ideas / What's Next

- Update analysis with: Newer 10-Ks, more 10-Ks, and 10-Qs
- Automate the import of data and creation of reports
- Regressions and correlation on risk factor language and economic performance

Discussion on SEC Data Scraping

The SECs organizes it's data labyrinth on the company-level data with a unique ID called the CIK (Central Index Key). The CIK is a 10 digit number with several leading 0s. On certain webpages and SEC texts, the leading 0s will be dropped and the number will be less than 10 digits, which may initially cause confusion.

The URL request structure of the 10-K Results Page is easy to replicate en masse for any given company once we have the CIKs. CIKs for all DOW companies can be found easily enough online. We found ours via a large list of CIKs on Kaggle.

The below URL structure stays the same from company to company, and the only thing that needs to be altered is the 10-digit CIK nested within the URL – '*&ciks=**0000731766***' – this can be done with a list of CIKs and some text joining in Excel. The '*forms=10-K*' section is an additional request parameter that filters the results to only yield 10-Ks. There are additional filtering parameters that could be identified by submitting searches with different filters and analyzing how the URL changes.

https://www.sec.gov/edgar/search/#/category=custom&ciks=0000731766&forms=10-K

Now that we have a list of 10-K search result URLs for all 30 DOW Companies, we want to extract the URLs of the first two 10-Ks for each company. We can iterate through the URLs using Selenium in Python. Make sure you have the compatible versions of Chrome and Chromedriver Installed. The below code opens each URL in Chrome, clicks the preview file button, opens the HTML document for the filing, then copies the URL and saves it to a list object, rinse and repeat for each URL. The loop was run twice for the first and second 10-Ks in each search result. This particular iteration opens the second most recent 10-K, but can be modified to gather the most recent 10-K URL as well.

```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time

# List of URLs
urls = [

  ### Paste our List of URLs here ###
]


# Array to store the final URLs
final_urls = []

# Loop through the URLs
for url in urls:

    # Open Chrome window
    driver = webdriver.Chrome()

    # Open the URL
    driver.get(url)

    # Wait for the page to load and for the element to be present
```

```
    time.sleep(5)


    try:
        elements = driver.find_elements(By.CLASS_NAME, "preview-file")
        if  len(elements) >= 2:
            # Click the second element
            elements[1].click()
            time.sleep(2)  # Wait for the next element to be clickable

        element2 = driver.find_element(By.CSS_SELECTOR, ".btn-warning")
        element2.click()

        # Wait for new tab to open
        time.sleep(5)

        # Switch to the new tab if it exists
        if len(driver.window_handles) > 1:
            new_tab = driver.window_handles[1]
            driver.switch_to.window(new_tab)
            final_urls.append(driver.current_url)
            driver.quit()

        else:
            print(f"No new tab opened for URL {url}")

        driver.quit()

    except Exception as e:
        print(f"Error occurred with URL {url}: {e}")
        driver.quit()


print(final_urls)
```

There are many ways to gather these URLs, many probably better than this method, but it worked nonetheless. One other method would be using the official SEC API to identify document submissions by company, then constructing URLs from official JSON API response data. See another wonderful example of SEC data gathering here.

Once the .htm links of the actual 10-K filings have been gathered, we *should* be able to scrape text from them using *requests* and *BeautifulSoup* in Python. Every properly-filed 10-K will have an Item '1A. Risk Factors' section, most 10-Ks will have a table of contents that point to sections, many will have hyperlinks in the table that bookend the sections. Complexity arises from companies who don't follow the standard format as there is no precise mandate to adhere to, thus more advanced parsing algorithms may need to be employed.

The other issue is the need to declare *who* we are when scraping SEC data. Upon attempting a test scrape of the first URL with BeautifulSoup, the following error message was issued:

"Your request has been identified as part of a network of automated tools outside of the acceptable policy and will be managed until action is taken to declare your traffic. Please declare your traffic by updating your user agent to include company specific information."

In the future, we can use this official method, but need to take special care to adhere to SEC requirements. In lieu of using the somewhat esoteric SEC API, there is a 3rd Party API that greatly eases the process and allows for 100 Free Requests with no credit card before a $55 monthly charge.

The below code is an interaction with the API using the 'requests' library that loops our 60 most recent 10-K document URLs as parameters in separate API calls and returns Item '1A. Risk Factors' for each as text. It puts them in a list that is then converted to a DataFrame with respective URLs as unique row keys.

```
import requests
import pandas as pd

# Your API key
api_key = ### Obtained API Key Here ###

# List of filing URLs
filing_urls = [

     ### Obtained Filing .htm URLs here ###
]

# Empty List for Results
results = []

# API endpoint
api_endpoint = 'https://api.sec-api.io/extractor'

# Loop through each URL and make the API request
for filing_url in filing_urls:
    params = {
        'url': filing_url,
        'item': '1A',
        'type': 'text',
        'token': api_key
    }

    response = requests.get(api_endpoint, params=params)

    if response.status_code == 200:
        item_1a_text = response.text
        results.append({'Source URL': filing_url, 'Item 1A Text': item_1a_text})
    else:
        print(f"Failed to fetch data for {filing_url}: {response.status_code} - {response.reason}

# Convert the list of dictionaries to a DataFrame
df = pd.DataFrame(results)
```

```
# Display the DataFrame
print(df)
```

From here, we perform other actions such removing line breaks, finding and replacing coded HTML character entities with their actual characters as well as joining in Ticker Symbols and 10-K dates to the DataFrame. Once we've gotten our data where we want it, we are ready to transition from Python to R and begin our text analyses.