A3_Victor_Huarniz

Victor Huarniz Noya

9/2/2021

Introduction to the case and data collection

As an accountant, my desire for including tech in our profession creating a "modern accounting" trend, I decided to analyze financial statements and tax reports. In this particular case, I will analyze the 10k of Alphabet Inc (Google) from 2016-2019.

The goal of this analysis is to see if we can use NLP to analyze these 140+ pages reports faster and focus on specific differences and analyze the storytelling of those. This would increase the efficiency of external financial audit reports where they can analyze their client's accounting reports. This can also help investors analyze specific differences from year to year which includes risks and external factors of the company, and ask about specific topics that might not be visible in a quick reading.

I chose Alphabet (Google) since I believe in them as leaders of technology development to contribute something remarkable in this new "modern accounting" topic.

Data collection

To analyze data, we have used the package "edgar" that lets us download any report submitted to the U.S. Securities and Exchange Commission (SEC). After dowloading their reports in html, we read them with read_html and proceeded to manipulate data as required for our analysis.

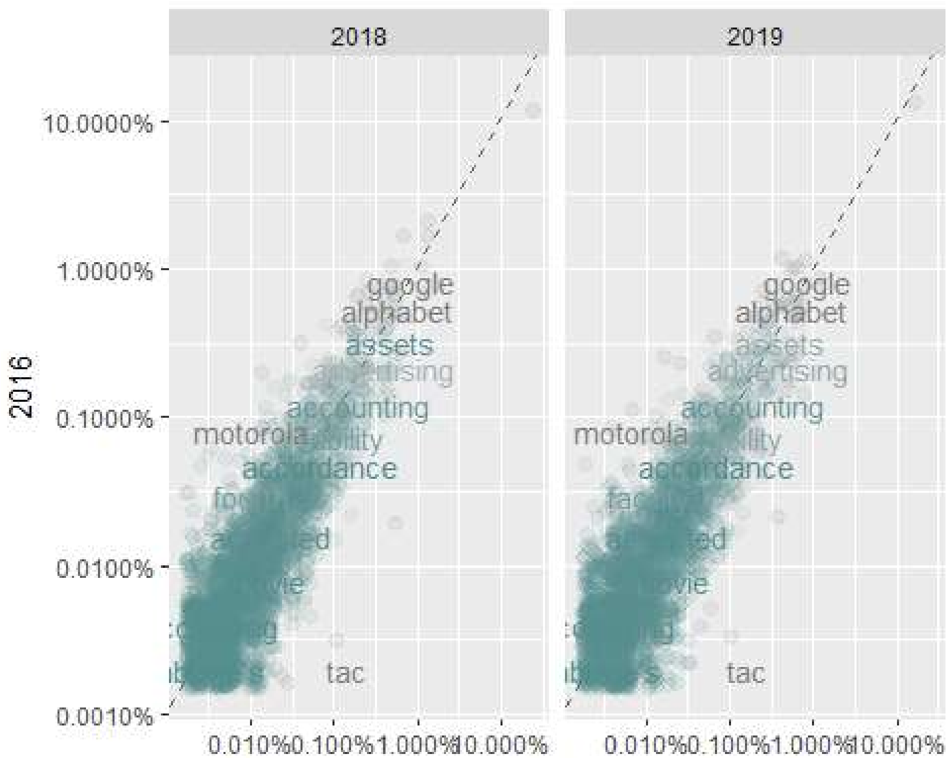
Our analysis consists on:

- 1 - Specific differences from one year to another: I compared each year's report to see any fundamental changes on the use of terms that could lead to a higher or change of importance. Do we have a specific update from a year to another?
- 2 - Key words used from 2016 to 2019: After comparing the last two years with a base year, I proceeded to analyze each year independently where I can find some key words that could lead me to specific topics to dig in for a better comprehension of each year. For this methodology, I deleted all numbers since we want to focus on keywords. Which key words are in Alphabet's audit reports?
- 3 - Related n-grams (2016-2019): Furthermore, I decided to analyze related bigrams (2-word groups) to plot a relationship net that interconnects them with key words. Can related n-grams tell us a story?

- 4 - Sentiments used in audit reports: Finally, I wanted to try getting smart clouds of sentiments using the words used in the reports. Can we get sentiments from the reports?

Analysis

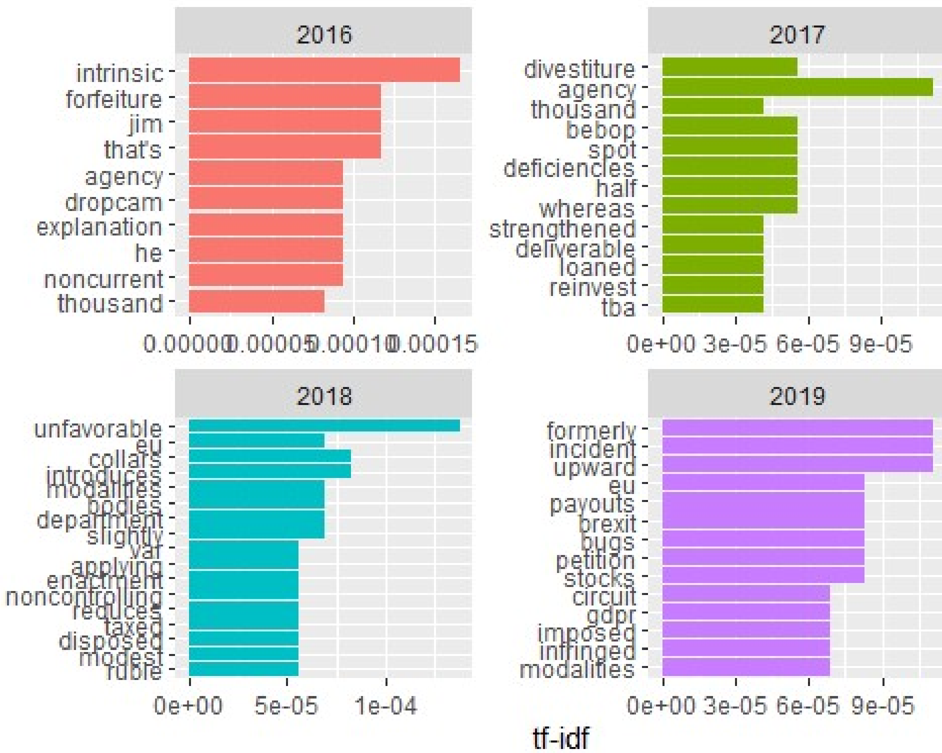
1 - Do we have a specific update from a year to another?



From these graphs we can conclude that in 2018 and 2019, Alphabet has talked more about properties and "tac" (traffic acquisition costs), getting a deep reading into these statements that both became important points to overwatch through time since TAC increased from $16 bn (38.9% of revenue) to $30 bn (44.4% of revenue) from 2016 to 2019, see Note of Cost of Revenues, and Google properties increased from $63 bn to $98 bn which is detailed to be one of the revenue performance keys for Alphabet. The changes in those are getting more detailed since they want to track the performance for each type and this might show a business core change as Amazon is changing to AWS, Alphabet might already changed to Google Properties.

2 - Which key words are in Alphabet's audit reports? (2016 - 2019)

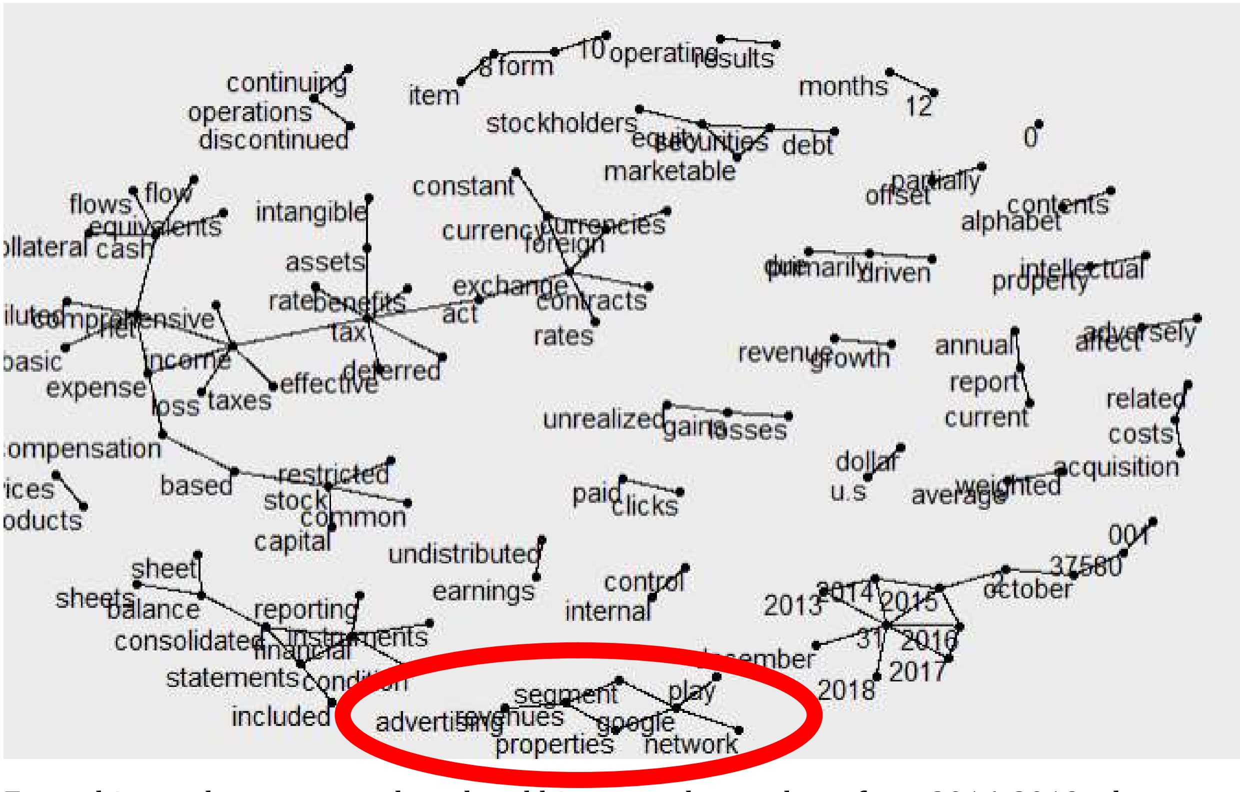
```
## Joining, by = "fiscal_year"

## Selecting by tf_idf
```



Using this analysis we can see some important words that are mentioned in the audit reports that had some influence in the company like “brexit” in 2019 which they detailed to “adversely affect their revenues and could subject them to new regulatory costs and challenges”. This risk is something fundamental to keep an eye on when referring to investments and the business operations.

3 - Can related n-grams tell us a story? (2016-2019)

```
##
## Attaching package: 'igraph'

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```



From this graph we can see the related biograms that we have from 2016-2019 where an important insight is that segment of revenues of Alphabet which is mainly related to google properties, googple play, and google network.

4 - Can we get sentiments from the reports?

```
## Loading required package: RColorBrewer

## Joining, by = "word"

##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths

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


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

As expected, we can see that the most popular sentiments are surprise and anticipation using the nrc evaluation and negative words using the bing evaluation, but we can also confirm certain keywords that are related to these sentiments. For example, we see words as drones, sabotage, and threatening that had been used in the reports. After a quick reading, we can confirm that the company indeed used these when referring to "their ongoing investments in safety, security, and content review will likely continue to identify abuse of our platforms and misuse of user data." This is actually an statement that the company reinforces that they have the ongoing resource to take care of these negative sentiments which are a real problem in our current digital environment.

Conclusion

We have seen that NLP can help with the accounting analysis of financial statements, it gives a quick storytelling of the company's performance identifying hot topics that can indeed be important to review in a directory meeting, with the CEO, CFO, and the investors.

Some insightful business decisions that can be made following these analysis are: * TAC and properties: Management should wisely follow a plan to hold the TAC increment, this may require an extra investment that can further reduce in a medium term these effects since the cost keeps increasing over time.

- Brexit: Brexit indeed is a delicate topic for Google and its operations in Europe. Since this is an external factor of the company, we might start negotiating with England about specific changes or try even to get a contract with the government to freeze current conditions to avoid instability for british operations.

- Google revenue units (google properties, google play, google network): This is an interesting output from the analysis. We have indeed seen that meanwhile AI is one the goals for google, actually google properties is one of the main current business core. So, either they can start increasing their comments for AI for the investors or focus on google properties as their most successfull current business core.

- Surprise, anticipation, and negative sentiments: We have seen that these hot topics are actually handled by the company, but they don't mention much positive things that the company develops. Instead, they focus on how they can reduce negative stuff happening outside the company. Therefore, I would suggest they increase the positive descriptions in their reports to not only show their strengths to the investors, but also make them focus on these positive strengths.

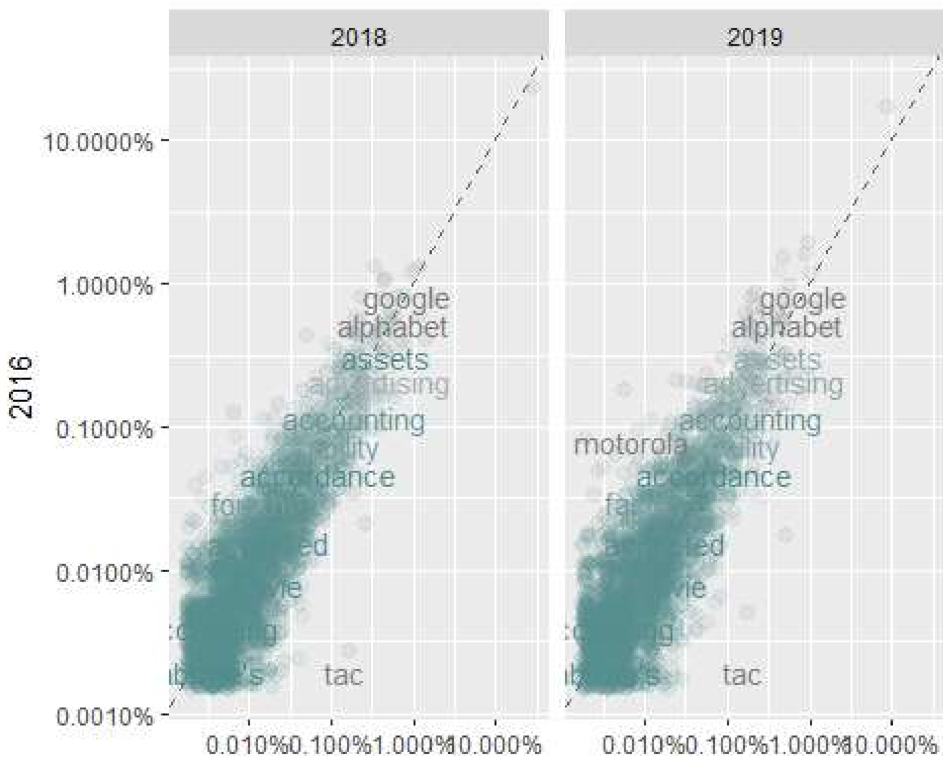
Appendix

```
#-----------------------------------------------------------{r data}
library(tidytext)
library(textreadr)
library(dplyr)
# Alphabet cik number 0001652044 Source:
https://www.sec.gov/edgar/searchedgar/cik.htm
# Edgar documentation https://cran.r-project.org/web/packages/edgar/edgar.pdf
```

```r
#install.packages("edgar") # install edgar to read sec APIs
library(edgar) #reading edgar library to use SEC data

#downloading data from SEC using Edgar package
output <- getFilingsHTML(cik.no = 0001652044, '10-K',
c(2016,2017,2018,2019),quarter=4)

## Downloading fillings. Please wait...
##   |
|                                                                      |   0%
|
|==================                                                    |  25%
|
|===================================                                   |  50%
|
|====================================================                  |  75%
|
|======================================================================| 100%
## Scrapping full EDGAR and converting to HTML...
##   |
|                                                                      |   0%
|
|==================                                                    |  25%
|
|===================================                                   |  50%
|
|====================================================                  |  75%
|
|======================================================================| 100%
## HTML filings are stored in 'Edgar filings_HTML view' directory.

#setting working path
setwd("C:/Users/victo/OneDrive/Escritorio/c/_Assignment/Edgar filings_HTML
view/Form 10-K/1652044")

#list of files
nm <- list.files(path="C:/Users/victo/OneDrive/Escritorio/c/_Assignment/Edgar
filings_HTML view/Form 10-K/1652044")

#setting a list with full paths of the files
files_paths <- paste("C:/Users/victo/OneDrive/Escritorio/c/_Assignment/Edgar
filings_HTML view/Form 10-K/1652044/",nm, sep="")


#using read document to import the data:
sec_2016 <- read_html(file=files_paths[1]) #This comes out as a vector
sec_2016 <- paste(sec_2016, collapse = " ") # This will give us a
concatenated vector

sec_2017 <- read_html(file=files_paths[2]) #This comes out as a vector
```

```r
sec_2017 <- paste(sec_2017, collapse = " ") # This will give us a
concatenated vector

sec_2018 <- read_html(file=files_paths[3]) #This comes out as a vector
sec_2018 <- paste(sec_2018, collapse = " ") # This will give us a
concatenated vector

sec_2019 <- read_html(file=files_paths[4]) #This comes out as a vector
sec_2019 <- paste(sec_2019, collapse = " ") # This will give us a
concatenated vector

#tokenizing

data(stop_words) # stop words

# creating tidy versions with tokens
df_2016 <- data_frame(line=1:length(sec_2016), text=sec_2016)

df_2016_tokens <- df_2016 %>%
                  unnest_tokens(word, text)%>%
                  anti_join(stop_words)

## Joining, by = "word"

df_2017 <- data_frame(line=1:length(sec_2017), text=sec_2017)

df_2017_tokens <- df_2017 %>%
                  unnest_tokens(word, text)%>%
                  anti_join(stop_words)

## Joining, by = "word"

df_2018 <- data_frame(line=1:length(sec_2018), text=sec_2018)

df_2018_tokens <- df_2018 %>%
                  unnest_tokens(word, text)%>%
                  anti_join(stop_words)

## Joining, by = "word"

df_2019 <- data_frame(line=1:length(sec_2019), text=sec_2019)

df_2019_tokens <- df_2019 %>%
                  unnest_tokens(word, text)%>%
                  anti_join(stop_words)

## Joining, by = "word"

df_2016_2019 <- bind_rows(mutate(df_2016, fiscal_year= "2016"),
                       mutate(df_2017, fiscal_year= "2017"),
                       mutate(df_2018, fiscal_year= "2018"),
```

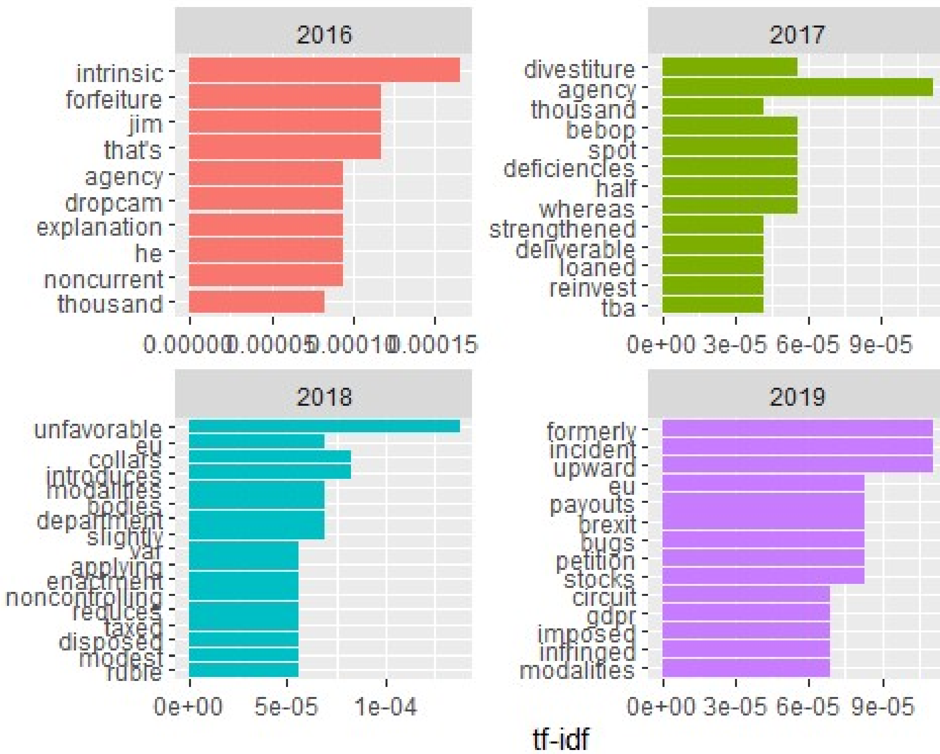
```r
                       mutate(df_2019, fiscal_year= "2019")
                       )

#---------------------------------------------------------{r analysis1}

library(tidyr)
library(stringr)

# merging tidy dataframes by year and getting the proportions
frequency <- bind_rows(mutate(df_2016_tokens, fiscal_year= "2016"),
                       mutate(df_2017_tokens, fiscal_year= "2017"),
                       mutate(df_2018_tokens, fiscal_year= "2018"),
                       mutate(df_2019_tokens, fiscal_year= "2019")
                       )%>%#closing bind_rows
                mutate(word=str_extract(word, "[a-z']+")) %>%
                count(fiscal_year, word) %>%
                group_by(fiscal_year) %>%
                mutate(proportion = n/sum(n))%>%
                select(-n) %>%
                spread(fiscal_year, proportion) %>%
                gather(fiscal_year, proportion, `2018`, `2019`)



# plotting the correlograms:
library(scales)
library(ggplot2)
ggplot(frequency, aes(x=proportion, y=`2016`,
                      color = abs(`2016`- proportion)))+
  geom_abline(color="grey40", lty=2)+
  geom_jitter(alpha=.1, size=2.5, width=0.3, height=0.3)+
  geom_text(aes(label=word), check_overlap = TRUE, vjust=1.5) +
  scale_x_log10(labels = percent_format())+
  scale_y_log10(labels= percent_format())+
  scale_color_gradient(limits = c(0,0.001), low = "darkslategray4", high =
"gray75")+
  facet_wrap(~fiscal_year, ncol=2)+
  theme(legend.position = "none")+
  labs(y= "2016", x=NULL)
```




```
#--------------------------------------------------------{r analysis2}

original_df <- df_2016_2019 %>%
                  unnest_tokens(word, text) %>%
                  count(fiscal_year, word, sort=TRUE) %>%
                  ungroup()

total_words <- original_df %>%
                  group_by(fiscal_year) %>%
                  summarize(total=sum(n))

year_words <- left_join(original_df, total_words)

## Joining, by = "fiscal_year"

###################################################
################# TF_IDF ##########################
###################################################


year_words <- year_words %>%
 bind_tf_idf(word, fiscal_year, n)


year_words%>%
  filter(is.na(as.numeric(word)))
```

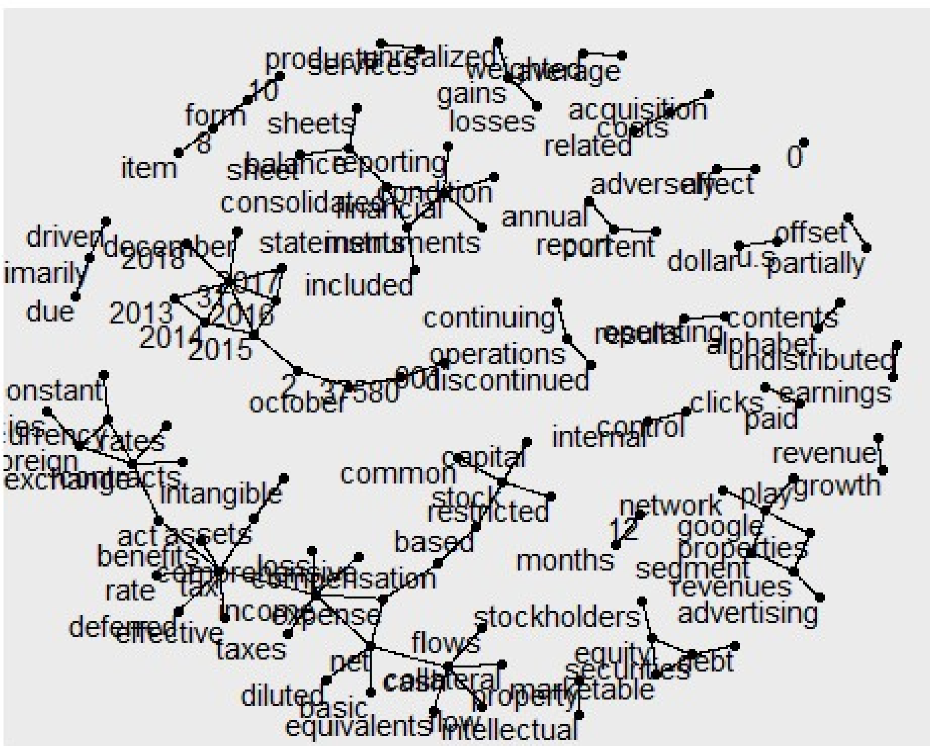
```
## # A tibble: 16,571 x 7
##    fiscal_year word      n total     tf   idf tf_idf
##    <chr>       <chr> <int> <int>  <dbl> <dbl>  <dbl>
##  1 2016        of     2441 58753 0.0415     0      0
##  2 2016        and    2333 58753 0.0397     0      0
##  3 2016        the    2139 58753 0.0364     0      0
##  4 2018        and    2021 50217 0.0402     0      0
##  5 2019        and    2015 50079 0.0402     0      0
##  6 2017        and    1977 49828 0.0397     0      0
##  7 2017        of     1961 49828 0.0394     0      0
##  8 2018        of     1937 50217 0.0386     0      0
##  9 2019        of     1918 50079 0.0383     0      0
## 10 2018        the    1878 50217 0.0374     0      0
## # ... with 16,561 more rows

#removing numbers

year_words %>%
  filter(is.na(as.numeric(word)))%>% # deleting numbers
  filter(is.na(as.numeric(gsub(",","",word))))%>% #deleting numbers with
commas
  arrange(desc(tf_idf))%>%
  filter(n<50)

## # A tibble: 13,325 x 7
##    fiscal_year word            n total       tf   idf   tf_idf
##    <chr>       <chr>       <int> <int>    <dbl> <dbl>    <dbl>
##  1 2019        impressions    11 50079 0.000220 1.39  0.000305
##  2 2016        split          10 58753 0.000170 1.39  0.000236
##  3 2019        fine           13 50079 0.000260 0.693 0.000180
##  4 2018        provisional    12 50217 0.000239 0.693 0.000166
##  5 2016        divestiture    14 58753 0.000238 0.693 0.000165
##  6 2016        intrinsic       7 58753 0.000119 1.39  0.000165
##  7 2017        kingdom        11 49828 0.000221 0.693 0.000153
##  8 2019        americas       11 50079 0.000220 0.693 0.000152
##  9 2019        apac           11 50079 0.000220 0.693 0.000152
## 10 2019        emea           11 50079 0.000220 0.693 0.000152
## # ... with 13,315 more rows

#what can we say about these words?

#############
# looking at the graphical apprach:
year_words %>%
  filter(is.na(as.numeric(word)))%>%
  filter(is.na(as.numeric(gsub(",","",word))))%>%
  arrange(desc(tf_idf)) %>%
  mutate(word=factor(word, levels=rev(unique(word)))) %>%
  group_by(fiscal_year) %>%
  filter(n<10)%>%
```

```
  top_n(10) %>%
  ungroup %>%
  ggplot(aes(word, tf_idf, fill=fiscal_year))+
  geom_col(show.legend=FALSE)+
  labs(x=NULL, y="tf-idf")+
  facet_wrap(~fiscal_year, ncol=2, scales="free")+
  coord_flip()

## Selecting by tf_idf
```




```
#-------------------------------------------------------{r analysis3}



sec_2016_2019_filtered <- df_2016_2019 %>%
                    unnest_tokens(bigram, text, token = "ngrams", n=2)%>%
                    separate(bigram, c("word1", "word2"), sep = " ") %>%
                    filter(!word1 %in% stop_words$word) %>%
                    filter(!word2 %in% stop_words$word) %>%
                    filter(!word1 == "NA")%>%
                    filter(!word2 == "NA")

sec_2016_2019_bigrams <- df_2016_2019 %>%
                    unnest_tokens(bigram, text, token = "ngrams", n=2)%>%
                    separate(bigram, c("word1", "word2"), sep = " ")%>%
                    filter(!word1 %in% stop_words$word) %>%
                    filter(!word2 %in% stop_words$word)%>%
                  count(word1, word2, sort = TRUE)
```

```
bigram_united <- sec_2016_2019_filtered %>%
                  unite(bigram, word1, word2, sep=" ") #we need to unite what
we split in the previous section

bigram_tf_idf <- bigram_united %>%
                  count(fiscal_year,bigram) %>%
                  bind_tf_idf(bigram, fiscal_year, n) %>%
                  arrange(desc(tf_idf))

library(igraph)
library(ggraph)

bigram_graph <- sec_2016_2019_bigrams %>%
                  filter(n>50) %>% #smaller for less data
                  graph_from_data_frame()

ggraph(bigram_graph, layout = "fr") +
  geom_edge_link()+
  geom_node_point()+
  geom_node_text(aes(label=name), vjust =1, hjust=1)
```




```
#-------------{r analysis4}
library(wordcloud)

original_years <- df_2016_2019 %>%
```

```
  group_by(fiscal_year) %>%
  mutate(lienumber = row_number())%>%
  ungroup() %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  count(word, sort=T)

## Joining, by = "word"

library(reshape2)
original_years %>%
  inner_join(get_sentiments("nrc")) %>%
  count(word, sentiment, sort=TRUE) %>%
  acast(word ~sentiment, value.var="n", fill=0) %>%
  comparison.cloud(colors = c("grey20", "grey50"),
                   max.words=20, scale= c(1,0.1))

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


```
original_years %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort=TRUE) %>%
  acast(word ~sentiment, value.var="n", fill=0) %>%
  comparison.cloud(colors = c("grey20", "grey50"),
                   max.words=50, scale= c(1,0.1))

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


manipulate
forged
inconsistent
unlawful
mar
hedge
elimination
irrelevant
disruption
inadequate
distraction
disaster
lacks
limitations
drones
declines
falls
conflict
defective
hinder
violate
injury
defamation
slower
costly
decline
expire
split
issue
cloud
infringement
failing
breach
degrade
risky
unethical