Sentiment Analysis of 3 Different News Agency's Reports on "Coronavirus World Economic Impact"

Introduction

At the beginning of year 2020, coronavirus broke out in Wuhan, China, which cast devastating impact on world economy in multiple areas. As China plays a role as both importer and exporter to the rest of the world, and its share of the global economy has surged in recent 20 years, the viral outbreak has already caused a worldwide economic loss – interrupted supply chain and manufacturing, suspended operations in multinational companies and international trade, stagnant tourism, and commodity price slump, etc.

This article aims in analyzing the sentiments on "coronavirus world economic impact" among 3 major news agency's reports by using text analytics tools, such as word frequency analysis, sentiment analysis, TF-IDF framework, and bigram visualizations in RStudio.

1 Word Frequency Analysis

Exhibit 1 shows the most frequent words in all 3 reports. Apart from the common big words which are always mentioned in the coronavirus news report, "trade", "supply", "closed", "hit", "demand", "crisis", "production", "manufacturing", "gdp", "bank" show up most frequently in the reports. We can assume a lot of negative things happening currently. Coronavirus impact hit the economy of most parts of the world, which has already been escalating to a worldwide crisis. Chinese demand on the luxury products decreased largely, even the demand on gas slumped, as most people prefer staying at home to avoid being infected. Due to the low demand and to prevent more affected cases happening, most businesses closed their factories, thus the production line stopped. However, countries like Japan, South Korea, etc. heavily depend on Chinese manufacturing, especially for car, smartphone, and semiconductors. Therefore, supply chain has disrupted, which led to a fatal influence on the international trade between China and other countries. The People's Bank of China reduced the interest rate and injected huge amounts of cash into markets in order to relieve the pressure of banks and borrowers. Domestic consumption and confidence in investment are low, plus trade is getting stuck, China's GDP would suffer at least for the first quarter of 2020.

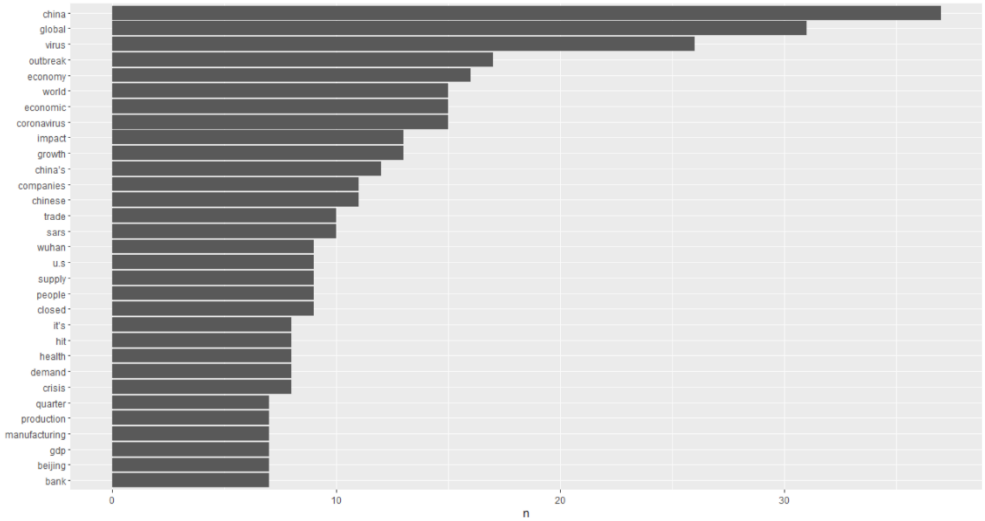


Exhibit 1: Most Frequent Words in 3 news reports

In Exhibit 2, we can clearly see that CNN and Bloomberg share more common words, as there are more words close to the fitted line, and the correlation score is 0.659, which is higher than 0.612, the score between CNN and TIME.

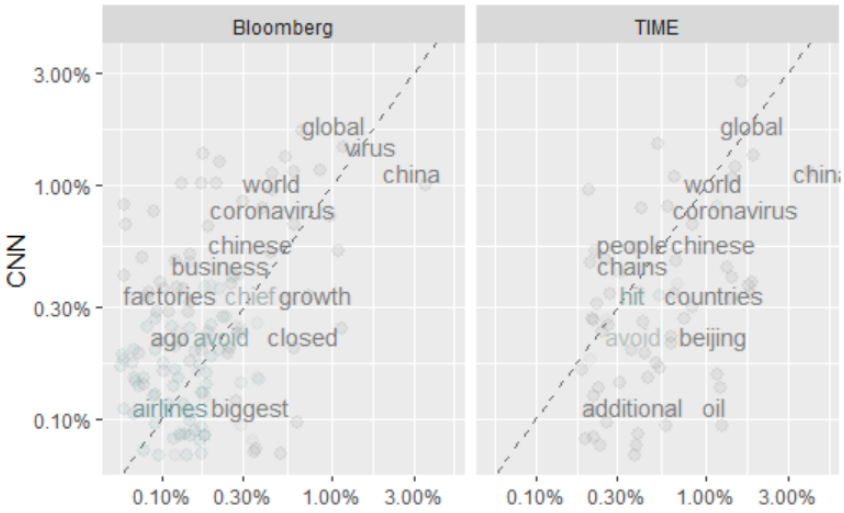


Exhibit 2: Correlograms between 3 news agencies reports

2 Sentiment Analysis

2.1 Word Cloud

Bing sentiment lexicon categorizes words in a binary way into positive and negative categories. Nrc sentiment lexicon is a list of words and their relations to eight basic emotions.

As we can see in Exhibit 2, there are much more negative words than positive ones. Except for the words that are used for describing the severeness of virus, “crisis”, “losses, “debt” are all negative words for economic recession. In Exihibit 4, more words are close to “anticipation” and “surprise”, and surprisingly, some words are even close to “positive”, so still nobody is sure about the future trend of the virus cases and economy conditions.

Exhibit 3: CNN Word Cloud Using “bing” Lexicon

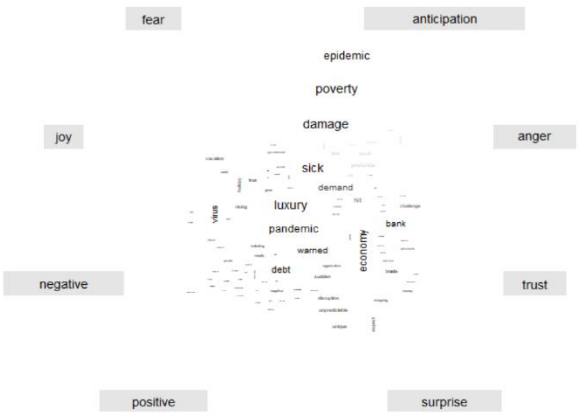


Exhibit 4: CNN Word Cloud Using “nrc” Lexicon

Although negative words still surpass positive ones in TIME word cloud using “bing” sentiment, interestingly, there are a bunch of words approach to the positive emotion by using “nrc” sentiment instead. Let’s put this finding aside first and leave it to the section of “comparison of the sentiment lexicons”.

Exhibit 5: TIME Word Cloud Using “bing” Lexicon

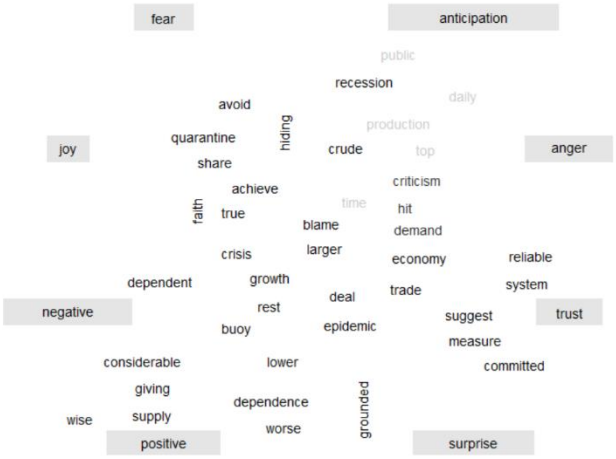


Exhibit 6: TIME Word Cloud Using “nrc” Lexicon

Based on the word clouds below, among the 3 news agencies, Bloomberg stands in a relatively neutral position on the issue of coronavirus world economic impact. The positive and negative words’ proportions are similar, and there’s no evident emotion expressed.

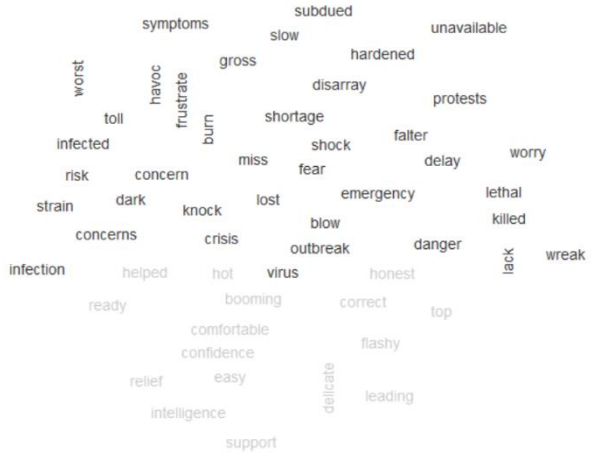


Exhibit 7: Bloomberg Word Cloud Using “bing” Lexicon

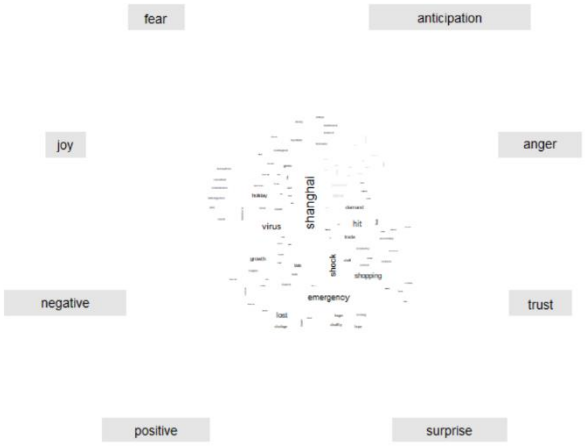


Exhibit 8: Bloomberg Word Cloud Using “nrc” Lexicon

Word cloud is good at showing an overview of the sentiments, but without scales, this framework still cannot be precise, and we cannot tell how strong each emotion is. Thus, we need to improve our framework.

2.2 Comparison of Sentiment Lexicons

From Exhibit 9 and Exhibit 10, we can see that by using “bing”, it shows that all three news agencies are using the negative sentiment to look at the issue of coronavirus impact on world economy, whereas “nrc” is much more positive. Among 3 agencies, TIME has the highest positive sentiment, which has already reached +10, and CNN has the highest negative sentiment, which has got to -60.

Exhibit 9: Positive and Negative Sentiment Analysis by Using “bing”

Exhibit 10: Positive and Negative Sentiment Analysis by Using “nrc”

Comparing three sentiment lexicons in each news agency’s report, we can still easily find that only NRC leads to positive sentiment, and among all, TIME report has the highest positive score. But why its score is so high, let’s look at how many positive and negative words are in these three lexicons.

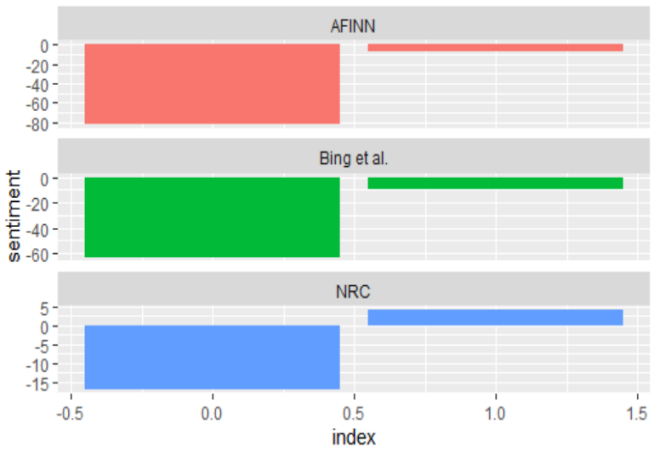


Exhibit 11: Three Lexicons Comparison in CNN report

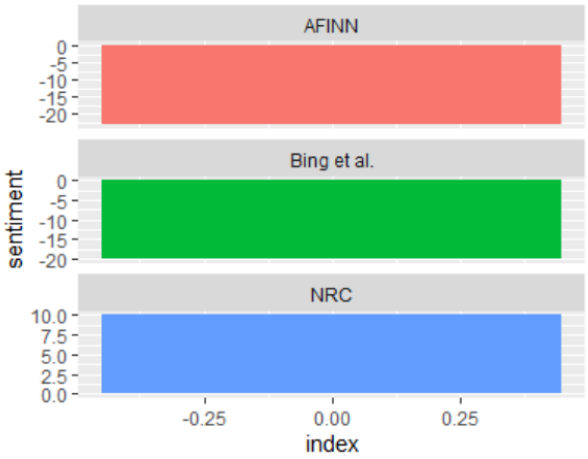


Exhibit 12: Three Lexicons Comparison in TIME report

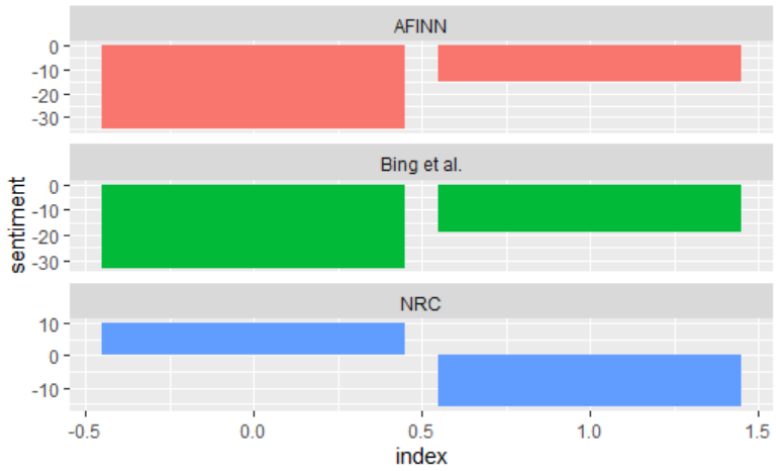


Exhibit 13: Three Sentiments Comparison in Bloomberg report

2.2 Word's Contribution to Sentiment

In all 3 reports, negative words appear far more frequently than positive ones. Besides, except for the common big words, like "virus" and "outbreak", "crisis" shows up in all 3 reports, which means viral outbreak cast really bad impact on economy.

Exhibit 14: Word Contribution to Sentiment in CNN report by using "bing"

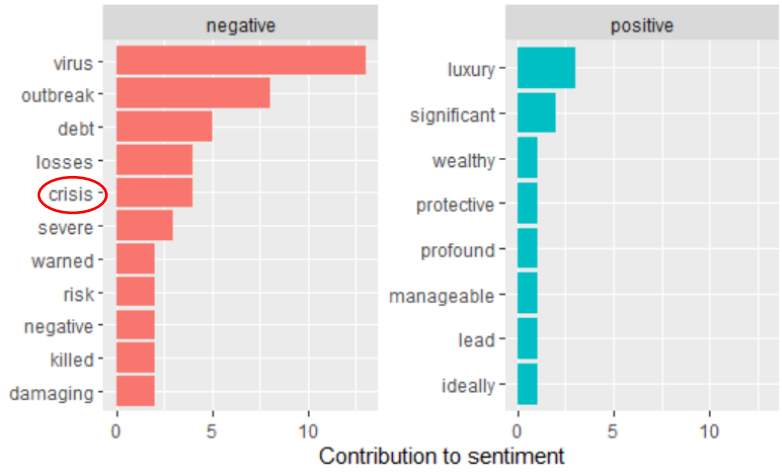


Exhibit 15: Word Contribution to Sentiment in TIME report by using "bing"

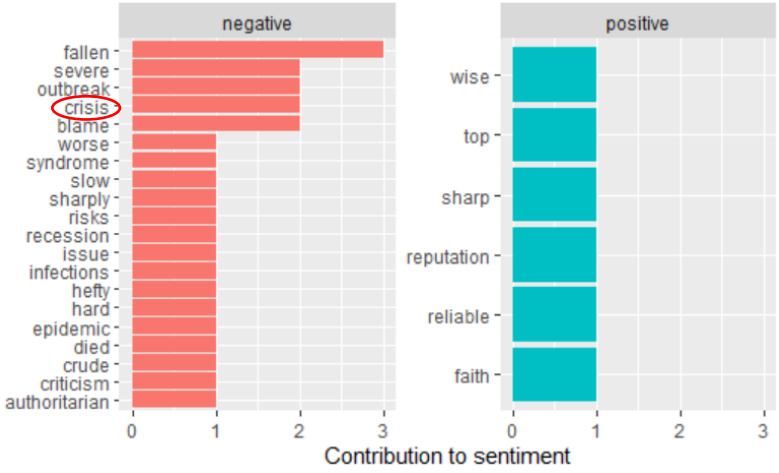
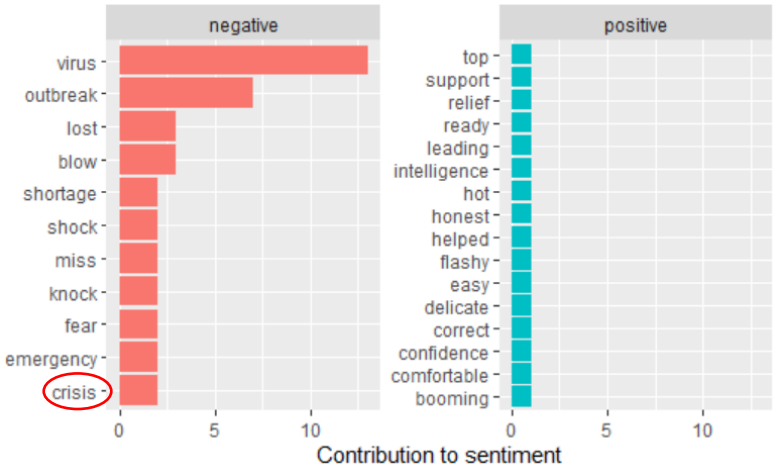


Exhibit 16: Word Contribution to Sentiment in Bloomberg report by using “bing”



In CNN report, among the eight emotions in NRC lexicon dictionary, “negative”, “trust”, “fear”, and “sadness” emotions are more dominant than others, which is quite different from using word cloud framework. Thus, it again emphasizes that the word cloud without scale has flaws in analyzing sentiments.

We should pay more attention in the emotion “trust” here, as coronavirus breakout brings us all bad results. Digging deeper, we can see that “bank”, “president”, “united”, “organization”, “optimist”, and “cooperation” are showing up here, which mean Chinese local government and world organizations are cooperating together to find active methods to fight with this economic crisis led by the virus. In addition, it’s because of this “trust” emotion, NRC leads to a relatively high positive score in the previous analysis.

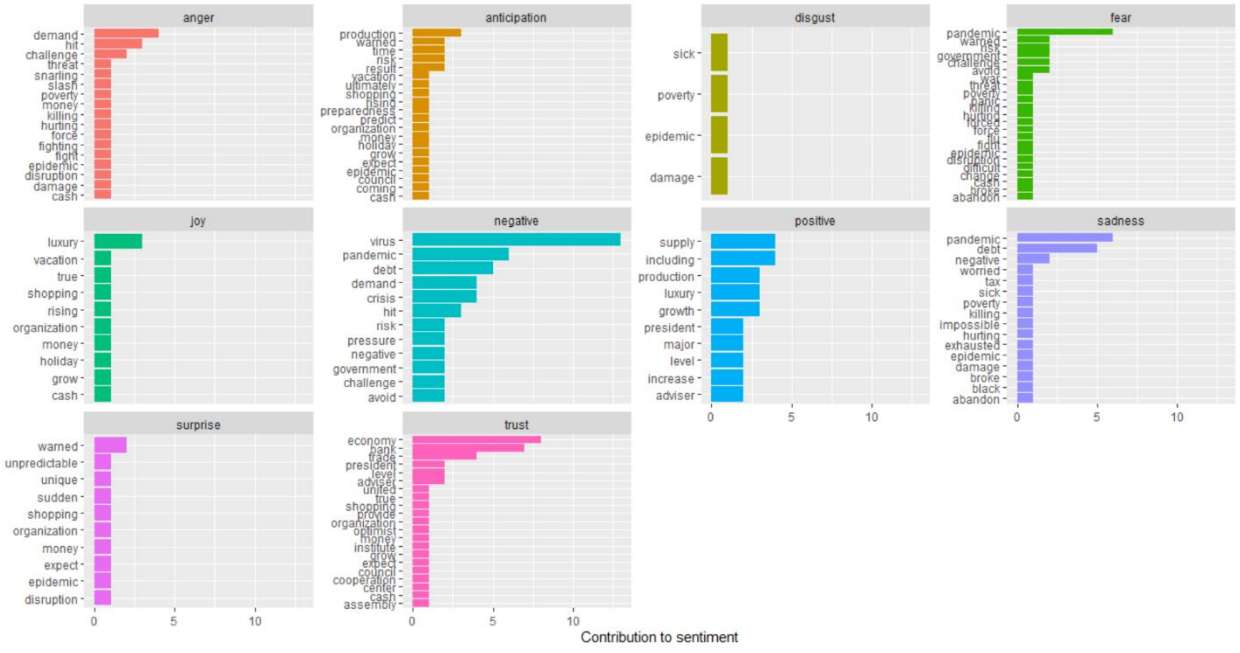


Exhibit 17: Word Contribution to Sentiment in CNN report by using “nrc”

In TIME report, among the eight emotions in NRC lexicon dictionary, “trust”, “positive”, “anticipation”, and “negative” emotions are more dominant than others, which can well explain why the NRC score of TIME is the highest among all three agencies.

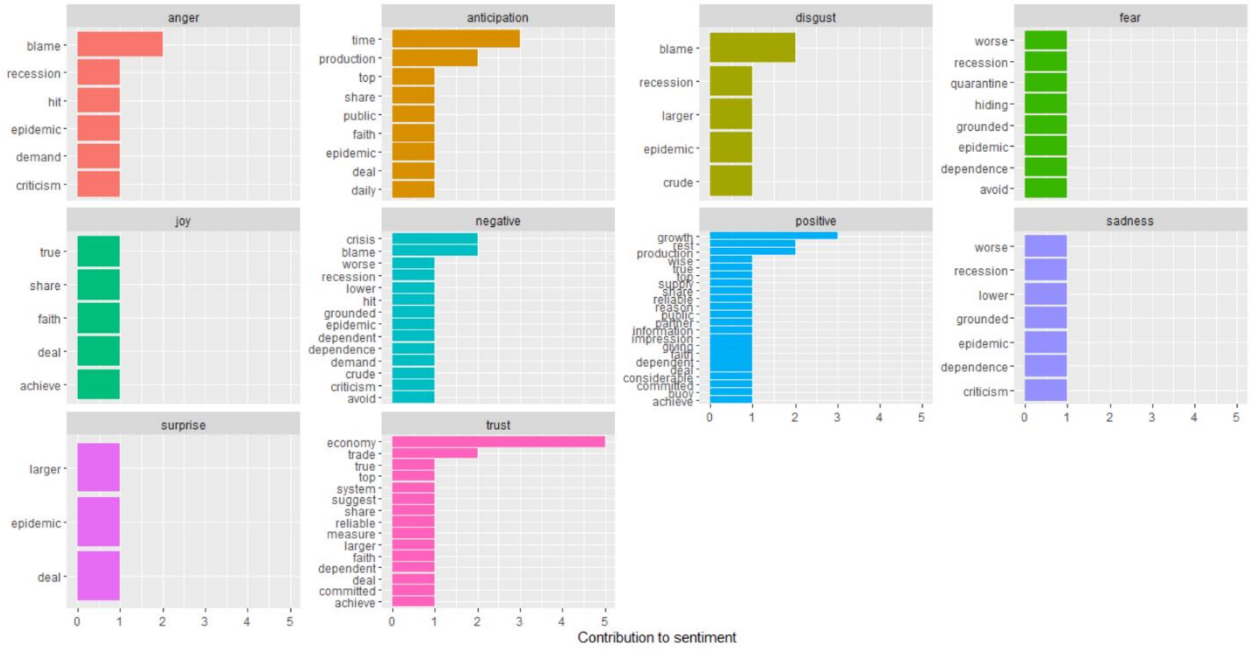


Exhibit 18: Word Contribution to Sentiment in TIME report by using “nrc”

For Bloomberg, both “negative” and “positive” emotions have the most frequent words, so it’s safe to say that Bloomberg is the most netrual agency on this virus topic.

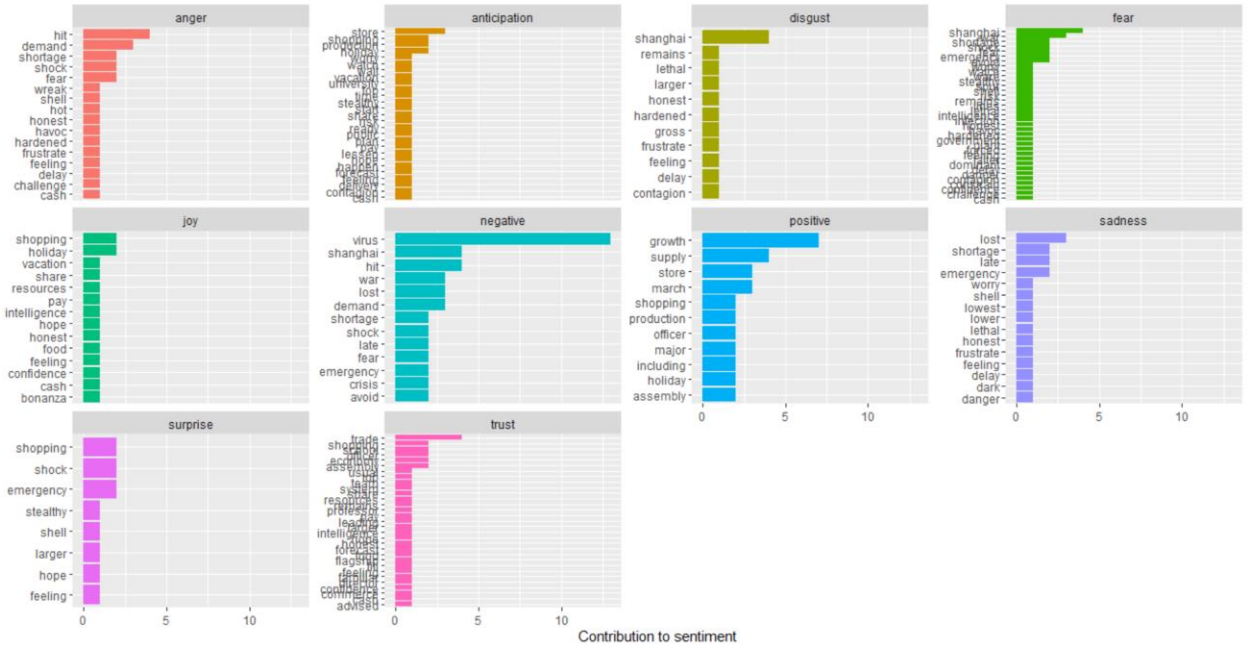


Exhibit 19: Word Contribution to Sentiment in Bloomberg report by using “nrc”

3 TF-IDF Framework

3.1 ZIPF's Law

This graph again shows that CNN and Bloomberg have strong correlation with each other. The term frequency (TF) of TIME is the highest among all. However, the news agency's report with the highest TF score doesn't mean that it's the best. We need to consider term frequency (TF) and inverse document frequency (IDF) at the same time.

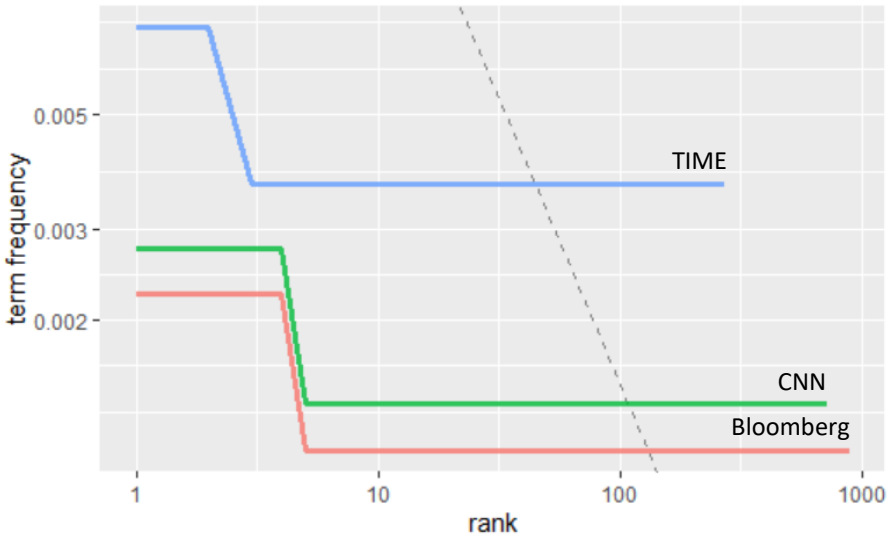


Exhibit 20: Term Frequency vs. Rank

3.2 TF-IDF

TF-IDF is a numeric statistic that is intended to reflect how important a word is to a document.

After calculating idf, and multipling with tf, we get the tf_idf score. Surprisingly, Bloomberg which has the lowest tf score, raises to the 1st rank of tf_idf score. "answer" and "honest" take the leading rank in the list.

```
# A tibble: 1,878 x 7
    line word            n news         tf   idf tf_idf
   <int> <chr>       <int> <chr>     <dbl> <dbl>  <dbl>
 1   104 answer          1 Bloomberg 0.5    4.64  2.32
 2   104 honest          1 Bloomberg 0.5    4.64  2.32
 3    96 beginning       1 Bloomberg 0.167  4.64  0.774
 4    96 breathe         1 Bloomberg 0.167  4.64  0.774
 5    96 relief          1 Bloomberg 0.167  4.64  0.774
 6    96 sigh            1 Bloomberg 0.167  4.64  0.774
 7   100 geopolitical    1 Bloomberg 0.167  4.64  0.774
 8   100 handle          1 Bloomberg 0.167  4.64  0.774
 9   100 lot             1 Bloomberg 0.167  4.64  0.774
10   100 nimbly          1 Bloomberg 0.167  4.64  0.774
# ... with 1,868 more rows
```

Exhibit 21: tf_idf rank among 3 reports

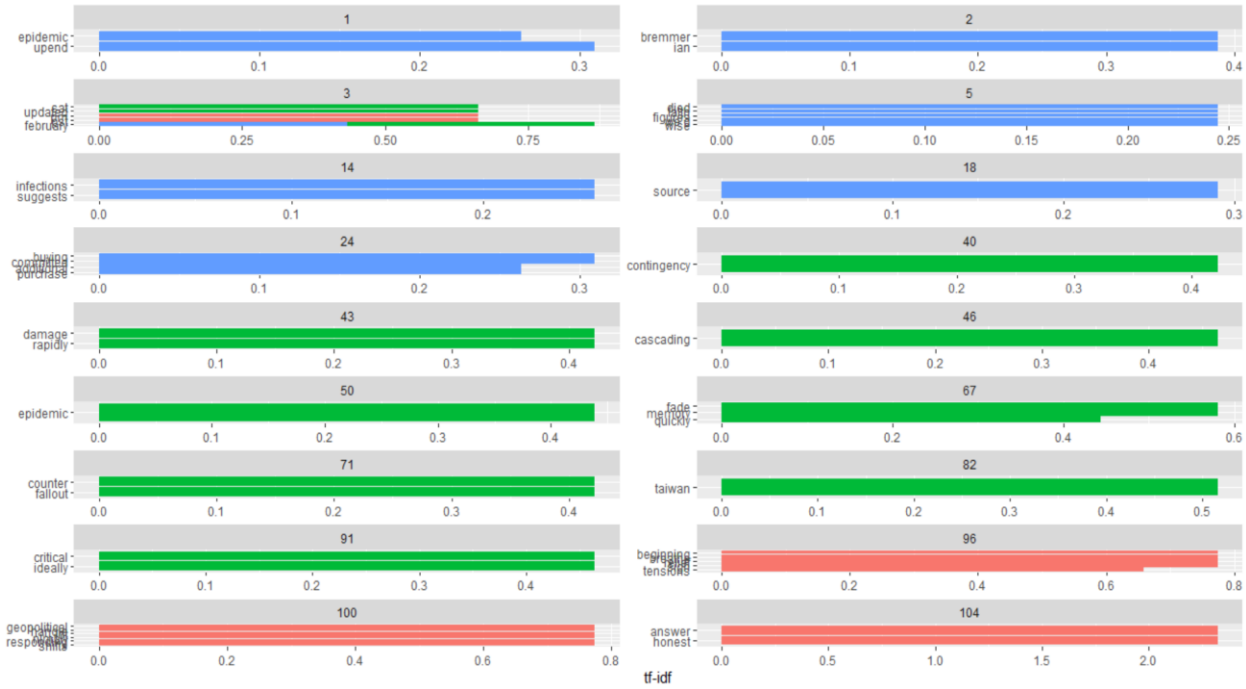


Exhibit 22: Visualization of tf_idf rank among 3 reports

4 Bigram Network

Although some words contribute to positive sentiments while others can be categorized into negative sentiments, they are still biased. As text analytics tools split the words one by one, only by combining the word with each other, can we evaluate the real sentiment that the author has. – here come the bigram network model.

In CNN bigram network, “China’s” and “gdp” have most connection with other words related to the economy.

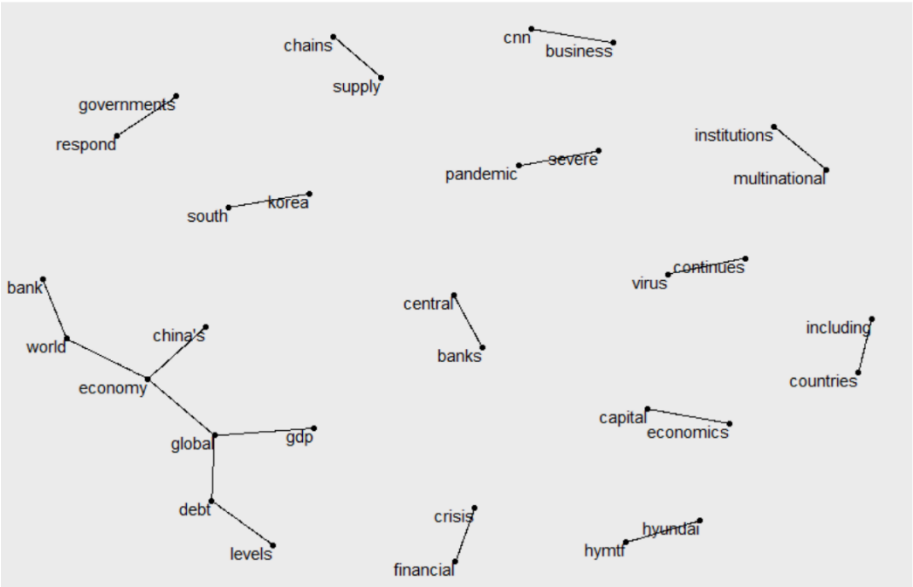


Exhibit 23: CNN Bigram Network

Bigram network of TIME news is quite sparse. Only 3 pairs of words have connections.

Exhibit 24: TIME Bigram Network

Bloomberg's bigram network is much better than Time's but still worse than CNN's, but we can see a long chain of "economy", "global", "supply" and "chain. This gives us an information that this viral outbreak's most severe impact on world economy is reflected by supply chain. As China is the second biggest world economy and one of the largest world factories, without enough output, many businesses in foreign countries will lose a lot of orders from customers.

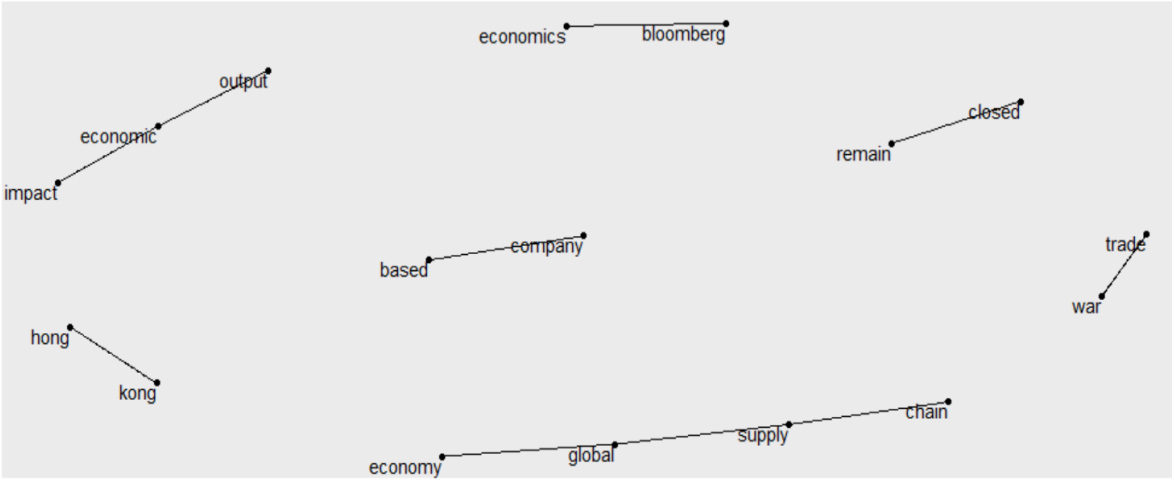


Exhibit 25: Bloomberg Bigram Network

Conclusion

Even though most people and countries have negative sentiments to the coronavirus outbreak, we still see the hope and trust behind the bad scenario. News agencies should take objective views on any diseases or disasters, in order to lead the public to positively fight any crisis!

Reference

[1] The coronavirus is already hurting the world economy. Here's why it could get really scary. Charles Riley and Julia Horowitz. CNN Business. February 8, 2020. Retrieved from https://www.cnn.com/2020/02/08/business/coronavirus-global-economy/index.html

[2] How the Coronavirus Epidemic Could Upend the Global Economy. IAN BREMMER. TIME. FEBRUARY 6, 2020. Retrieved from https://time.com/5778995/coronavirus-china-global-economy/

[3] The Coronavirus Is Infecting the Global Economy. Here's How. Cecile Daurat. January 31, 2020. Retrieved from https://www.bloomberg.com/news/articles/2020-01-31/the-global-economy-is-getting-infected-by-the-virus

Appendix

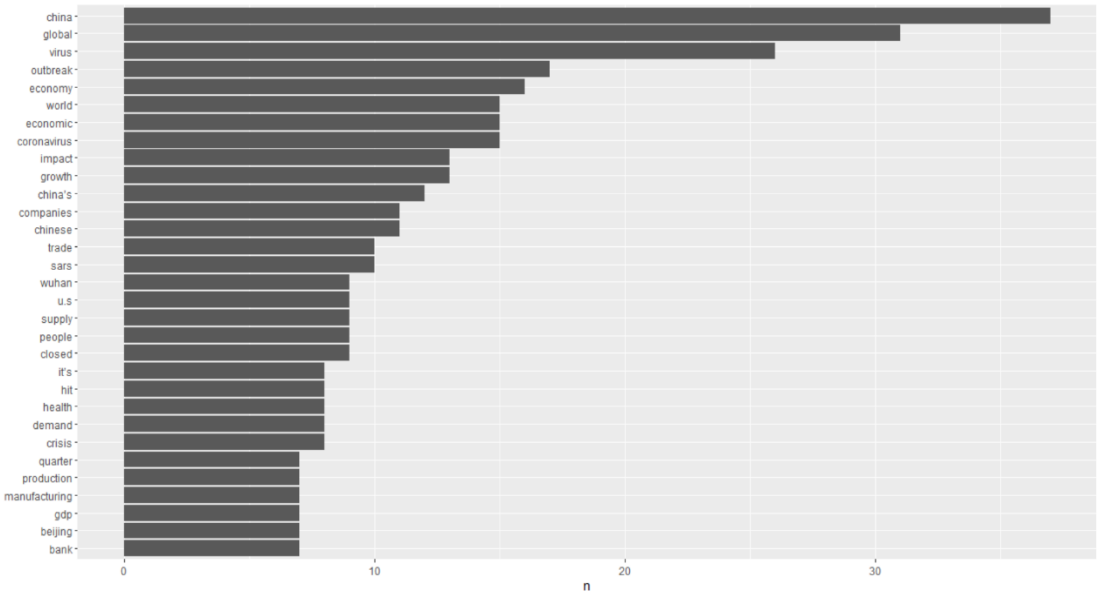
[Input]

```
# importing libraries

library(textreadr) # load documents

library(textdata)  # provide access to text-related data sets

library(tidyr)     # tidy the data

library(dplyr)     # data manipulation

library(stringr)   # extract words from a sentence

library(tidytext)  # text mining

library(tm)        # text mining

library(scales)    # scaling

library(reshape2)  # transform data between wide and long formats

library(wordcloud) # visualize the keyword as a word cloud

library(ggplot2)   # visualization tool

library(igraph)    # for bigram network

library(ggraph)    # for bigram network


# loading the .txt files

setwd('C:/Users/isabe/OneDrive/Desktop/MSBA-DD/Module B/Text Analytics/Individual Assignment')

CNN <- read_document('CNN.txt')

TIME <- read_document('TIME.txt')
```

```
BLB <- read_document('Bloomberg.txt')

# converting .txt files to dataframes
CNN_df <- tibble(line=1:95, text = CNN, stringsAsFactors=FALSE)
TIME_df <- tibble(line=1:35, text = TIME, stringsAsFactors=FALSE)
BLB_df <- tibble(line=1:104, text = BLB, stringsAsFactors=FALSE)

# remove nunerics in the text
CNN_df$text <- gsub("[0-9]+", "", CNN_df$text)
TIME_df$text <- gsub("[0-9]+", "", TIME_df$text)
BLB_df$text <- gsub("[0-9]+", "", BLB_df$text)

# toeknization
CNN_tokens <- CNN_df %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by="word") %>%
  count(line, word, sort=TRUE) %>%
  ungroup()
TIME_tokens <- TIME_df %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by="word") %>%
  count(line, word, sort=TRUE) %>%
  ungroup()
BLB_tokens <- BLB_df %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by="word") %>%
  count(line, word, sort=TRUE) %>%
  ungroup()
```

```
##############################################
############### Word Frequency ##############
##############################################
# combine three dataframes together
CNN_df$news <- 'CNN'
TIME_df$news <- 'TIME'
BLB_df$news <- 'BLB'
all_df <- rbind(CNN_df, TIME_df, BLB_df)

tidy_news <- all_df %>%
  group_by(news) %>%
  mutate(linenumber = row_number()) %>%
  ungroup() %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by="word") %>%
  count(word, sort = TRUE)

tidy_news %>%
  filter(n > 6) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```

[Output]

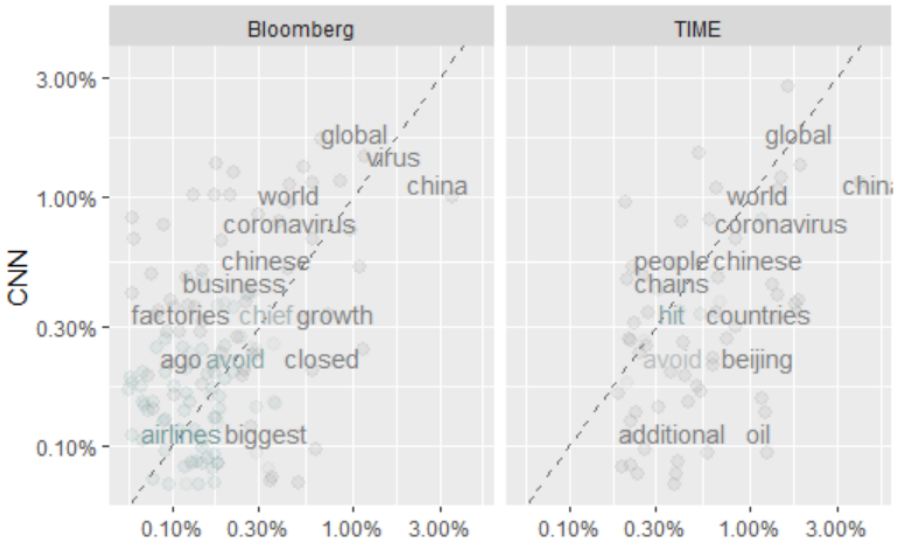


[Input]

```
# count the word frequency
frequency <- bind_rows(mutate(CNN_tokens, news="CNN"),
                       mutate(TIME_tokens, news="TIME"),
                       mutate(BLB_tokens, news="Bloomberg")
)%>%#closing bind_rows
  mutate(word=str_extract(word, "[a-z']+")) %>%
  count(news, word) %>%
  group_by(news) %>%
  mutate(proportion = n/sum(n))%>%
  select(-n) %>%
  spread(news, proportion) %>%
  gather(news, proportion, TIME, Bloomberg)


#let's plot the correlograms:
ggplot(frequency, aes(x=proportion, y=CNN,
                      color = abs(CNN- 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(~news, ncol=2)+
 theme(legend.position = "none")+
 labs(y= "CNN", x=NULL)
```

[Output]



[Input]

```
# correlation test
cor.test(data=frequency[frequency$news=="Bloomberg",],
      ~proportion + CNN)
```

[Output]

```
        Pearson's product-moment correlation

data:  proportion and CNN
t = 10.077, df = 132, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.5514932 0.7456167
sample estimates:
      cor
0.6594092
```

```
cor.test(data=frequency[frequency$news=="TIME",],
      ~proportion + CNN)
```

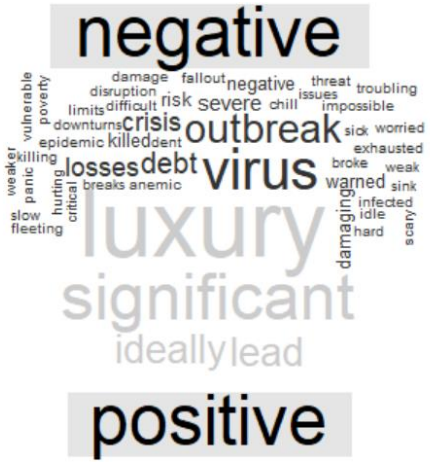
[Output]

```
        Pearson's product-moment correlation

data:  proportion and CNN
t = 6.0416, df = 61, p-value = 9.949e-08
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.4291454 0.7464586
sample estimates:
      cor
0.6118537
```

[Input]

```
#####################################################

############### Sentiment Analysis ###############

#####################################################

# word cloud

CNN_tokens %>%

  anti_join(stop_words, by="word") %>%

  inner_join(get_sentiments("bing"), by="word") %>% # binary

  count(word, sentiment, sort=TRUE) %>%

  acast(word ~sentiment, value.var="n", fill=0) %>%

  comparison.cloud(colors = c("grey20", "gray80"),

                   max.words=100)
```

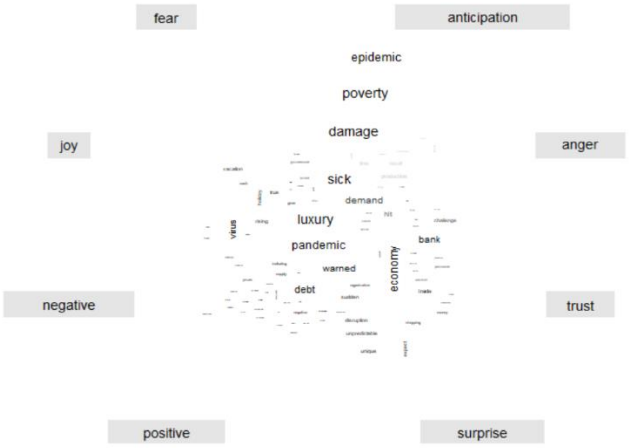
[Output]



[Input]

```
CNN_tokens %>%
  anti_join(stop_words, by="word") %>%
  inner_join(get_sentiments("nrc"), by="word") %>% # flavor
  count(word, sentiment, sort=TRUE) %>%
  acast(word ~sentiment, value.var="n", fill=0) %>%
  comparison.cloud(colors = c("grey20", "gray80"),
                   max.words=100, scale=c(0.9, 0,9),
                   fixed.asp=TRUE, title.size=0.9)
```

[Output]



[Input]

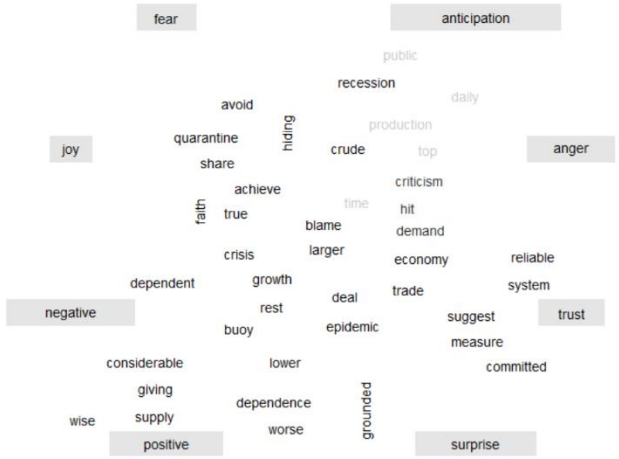
```
TIME_tokens %>%
anti_join(stop_words, by="word") %>%  inner_join(get_sentiments("bing"), by="word") %>% # binary
  count(word, sentiment, sort=TRUE) %>%
  acast(word ~sentiment, value.var="n", fill=0) %>%
  comparison.cloud(colors = c("grey20", "gray80"),
                   max.words=100, scale=c(0.9, 0.9),
                   fixed.asp=TRUE, title.size=0.9)
```

[Output]

[Input]

```
TIME_tokens %>%
  anti_join(stop_words, by="word") %>%
  inner_join(get_sentiments("nrc"), by="word") %>% # binary

count(word, sentiment, sort=TRUE) %>%
  acast(word ~sentiment, value.var="n", fill=0) %>%
  comparison.cloud(colors = c("grey20", "gray80"),
                   max.words=100, scale=c(0.9, 0.9),
                   fixed.asp=TRUE, title.size=0.9)
```

[Output]



[Input]

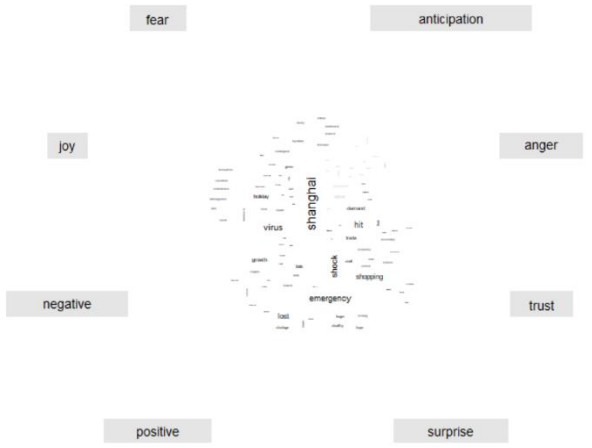
```
BLB_tokens %>%
  anti_join(stop_words, by="word") %>%
  inner_join(get_sentiments("bing"), by="word") %>% # binary
  count(word, sentiment, sort=TRUE) %>%
  acast(word ~sentiment, value.var="n", fill=0) %>%
  comparison.cloud(colors = c("grey20", "gray80"),
                   max.words=100, scale=c(0.9, 0.9),
                   fixed.asp=TRUE, title.size=0.9)
```

[Output]

[Input]

```
BLB_tokens %>%
  anti_join(stop_words, by="word") %>%
  inner_join(get_sentiments("nrc"), by="word") %>% # binary
  count(word, sentiment, sort=TRUE) %>%
  acast(word ~sentiment, value.var="n", fill=0) %>%
  comparison.cloud(colors = c("grey20", "gray80"),
                   max.words=100, scale=c(0.9, 0,9),
                   fixed.asp=TRUE, title.size=0.9)
```

[Output]



[Input]

```
# positive and negative sentiments among 3 news reports
tidy_news <- all_df %>%
  group_by(news) %>%
  mutate(linenumber = row_number()) %>%
  ungroup() %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by="word")

# using bing
news_sentiment <- tidy_news %>%
  inner_join(get_sentiments("bing"), by="word") %>%  # no positive sentiment
  count(news, index = linenumber %/% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)

ggplot(news_sentiment, aes(index, sentiment, fill = news)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~news, ncol = 2, scales = "free_y")
```

[Output]

[Input]

```
# using nrc
news_sentiment <- tidy_news %>%
  inner_join(get_sentiments("nrc"), by="word") %>%  #CNN and BLB still have some postive sentiments
  count(news, index = linenumber %/% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)


ggplot(news_sentiment, aes(index, sentiment, fill = news)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~news, ncol = 2, scales = "free_y")
```

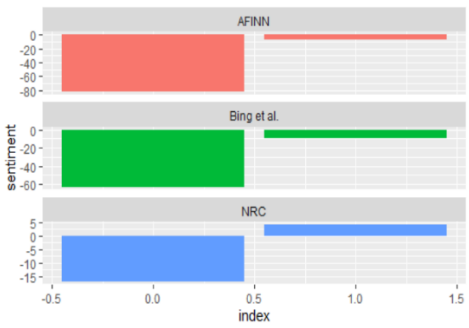
[Output]

[Input]

```
# comparing three sentiment lexicons in each news report seperately
CNN <- tidy_news %>%
  filter(news == "CNN")
```

```
afinn <- CNN %>%
  inner_join(get_sentiments("afinn"), by="word") %>%
  group_by(index = linenumber %/% 80) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "AFINN")

bing_and_nrc <- bind_rows(CNN %>%
                            inner_join(get_sentiments("bing"), by="word") %>%
                            mutate(method = "Bing et al."),
                          CNN %>%
                            inner_join(get_sentiments("nrc") %>%
                                         filter(sentiment %in% c("positive",
                                                                 "negative")), by="word") %>%
                            mutate(method = "NRC")) %>%
  count(method, index = linenumber %/% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)

bind_rows(afinn,
          bing_and_nrc) %>%
  ggplot(aes(index, sentiment, fill = method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")
```

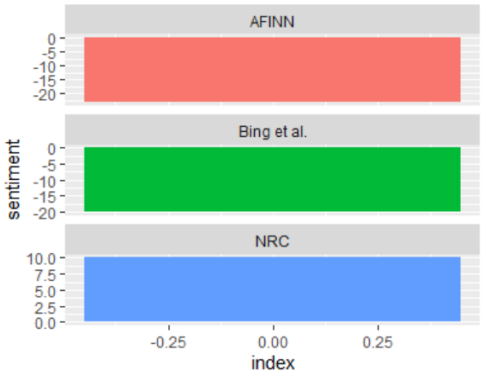
[Output]



[Input]

```
TIME <- tidy_news %>%
  filter(news == "TIME")

afinn <- TIME %>%
  inner_join(get_sentiments("afinn"), by="word") %>%
  group_by(index = linenumber %/% 80) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "AFINN")

bing_and_nrc <- bind_rows(TIME %>%
                          inner_join(get_sentiments("bing"), by="word") %>%
                          mutate(method = "Bing et al."),
                        TIME %>%
                          inner_join(get_sentiments("nrc") %>%
                                   filter(sentiment %in% c("positive",
                                                           "negative")), by="word") %>%
                          mutate(method = "NRC")) %>%
  count(method, index = linenumber %/% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)

bind_rows(afinn,
          bing_and_nrc) %>%
  ggplot(aes(index, sentiment, fill = method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")
```

[Output]

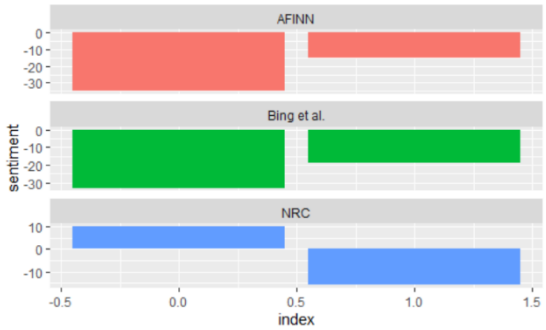


[Input]

```
BLB <- tidy_news %>%
  filter(news == "BLB")


afinn <- BLB %>%
  inner_join(get_sentiments("afinn"), by="word") %>%
  group_by(index = linenumber %/% 80) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "AFINN")


bing_and_nrc <- bind_rows(BLB %>%
                            inner_join(get_sentiments("bing"), by="word") %>%
                            mutate(method = "Bing et al."),
                          BLB %>%
                            inner_join(get_sentiments("nrc") %>%
                                         filter(sentiment %in% c("positive",
                                                                 "negative")), by="word") %>%
                            mutate(method = "NRC")) %>%
  count(method, index = linenumber %/% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)


bind_rows(afinn,
```

```
        bing_and_nrc) %>%
  ggplot(aes(index, sentiment, fill = method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")
```

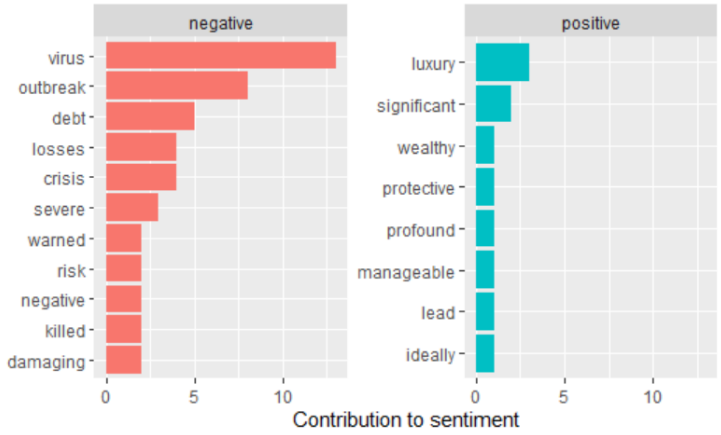
[Output]



[Input]

```
# Most common positive and negative words in each news report
CNN_bing_counts <- CNN_tokens %>%
  inner_join(get_sentiments("bing"), by="word") %>%
  count(word, sentiment, sort=T) %>%
  ungroup()


CNN_bing_counts %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word=reorder(word, n)) %>%
  ggplot(aes(word, n, fill=sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y")+
  labs(y="Contribution to sentiment", x=NULL)+
  coord_flip()
```

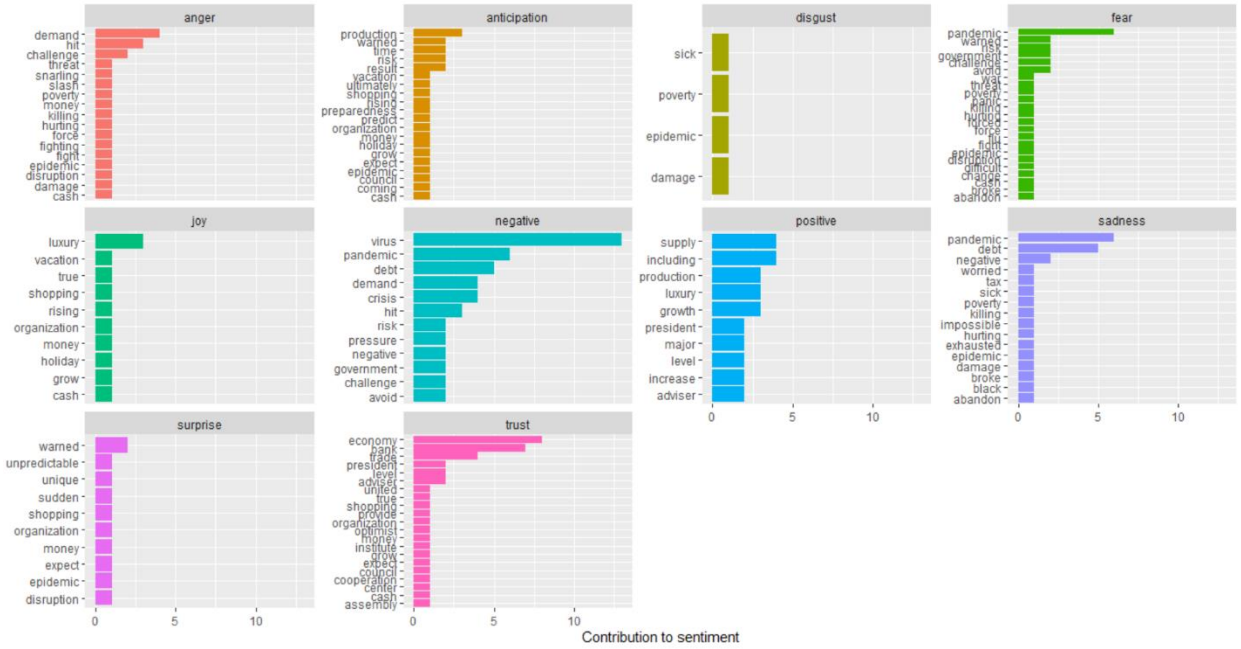
[Output]



[Input]

```
CNN_nrc_counts <- CNN_tokens %>%
  inner_join(get_sentiments("nrc"), by="word") %>%
  count(word, sentiment, sort=T) %>%
  ungroup()


CNN_nrc_counts %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word=reorder(word, n)) %>%
  ggplot(aes(word, n, fill=sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y")+
  labs(y="Contribution to sentiment", x=NULL)+
  coord_flip()
```

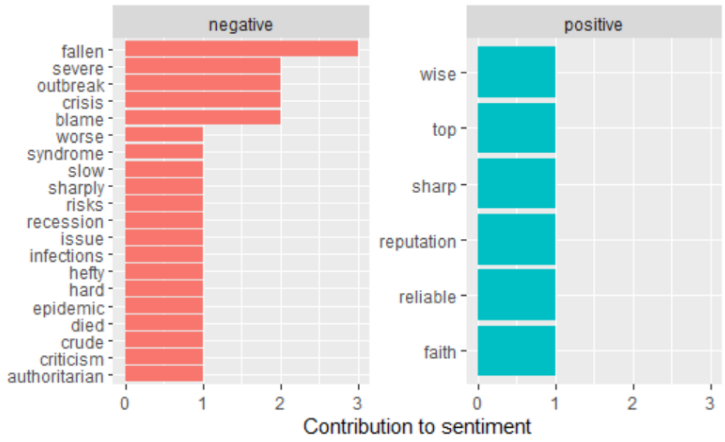
[Output]



[Input]

```
TIME_bing_counts <- TIME_tokens %>%
  inner_join(get_sentiments("bing"), by="word") %>%
  count(word, sentiment, sort=T) %>%
  ungroup()


TIME_bing_counts %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word=reorder(word, n)) %>%
  ggplot(aes(word, n, fill=sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y")+
  labs(y="Contribution to sentiment", x=NULL)+
  coord_flip()
```

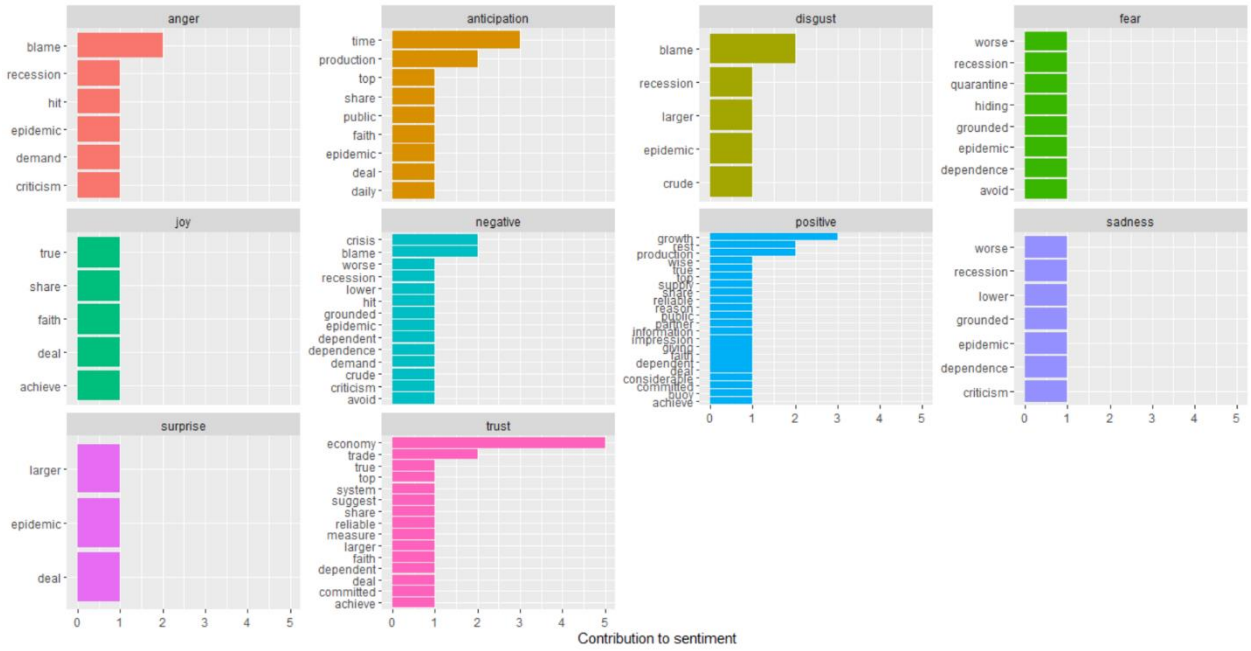
[Output]



[Input]

```
TIME_nrc_counts <- TIME_tokens %>%
  inner_join(get_sentiments("nrc"), by="word") %>%
  count(word, sentiment, sort=T) %>%
  ungroup()


TIME_nrc_counts %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word=reorder(word, n)) %>%
  ggplot(aes(word, n, fill=sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y")+
  labs(y="Contribution to sentiment", x=NULL)+
  coord_flip()
```

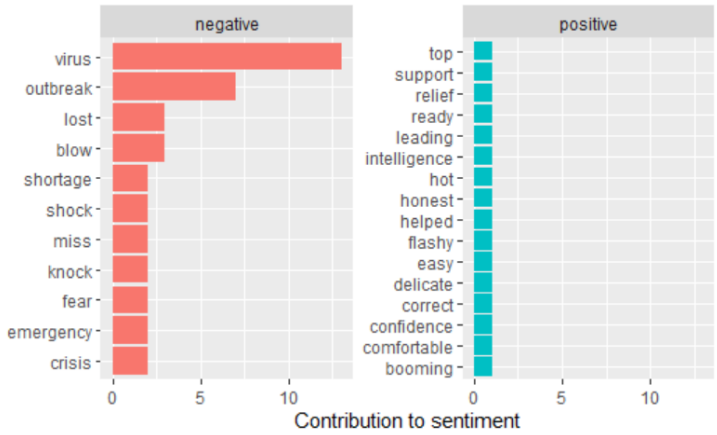
[Output]



[Input]

```
BLB_bing_counts <- BLB_tokens %>%
  inner_join(get_sentiments("bing"), by="word") %>%
  count(word, sentiment, sort=T) %>%
  ungroup()

BLB_bing_counts %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word=reorder(word, n)) %>%
  ggplot(aes(word, n, fill=sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y")+
  labs(y="Contribution to sentiment", x=NULL)+
  coord_flip()
```

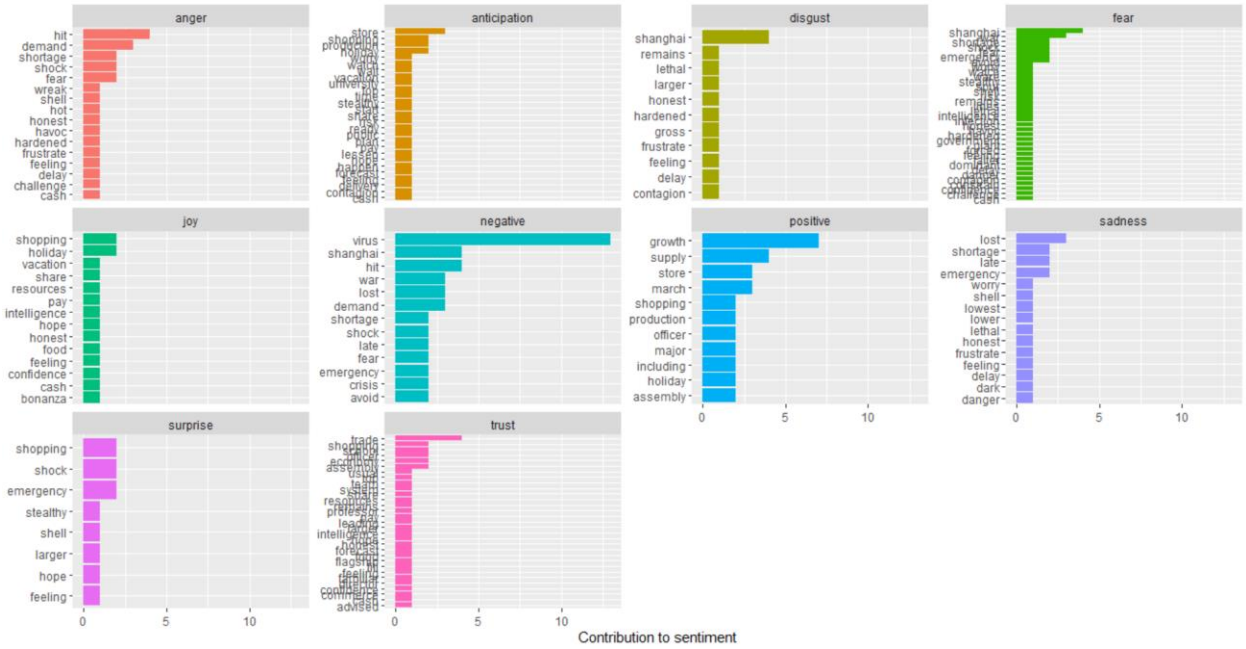
[Output]



[Input]

```
BLB_nrc_counts <- BLB_tokens %>%
 inner_join(get_sentiments("nrc"), by="word") %>%
 count(word, sentiment, sort=T) %>%
 ungroup()

BLB_nrc_counts %>%
 group_by(sentiment) %>%
 top_n(10) %>%
 ungroup() %>%
 mutate(word=reorder(word, n)) %>%
 ggplot(aes(word, n, fill=sentiment)) +
 geom_col(show.legend = FALSE) +
 facet_wrap(~sentiment, scales = "free_y")+
 labs(y="Contribution to sentiment", x=NULL)+
 coord_flip()
```

[Output]

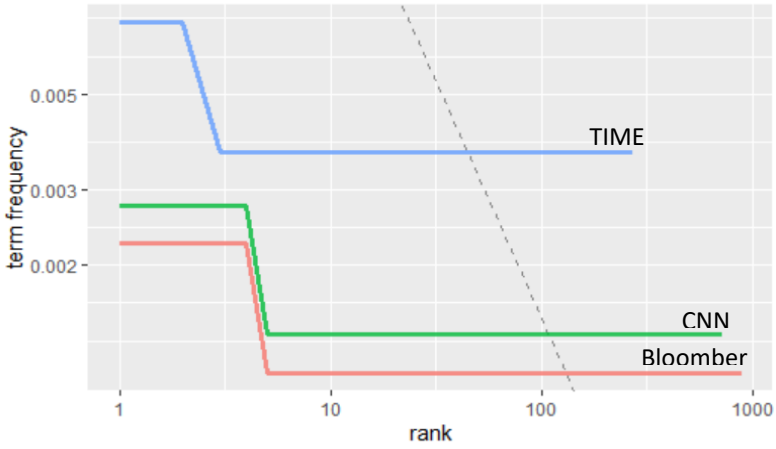


[Input]

```
################################

####### TF-IDF framework#######

################################


# ZIPF's law

all_tokens <- bind_rows(mutate(CNN_tokens, news="CNN"),

                        mutate(TIME_tokens, news="TIME"),

                        mutate(BLB_tokens, news="Bloomberg"))


freq_by_rank <- all_tokens %>%

  group_by(news) %>%

  mutate(rank = row_number(),

         `term frequency` = n/sum(n))


#plotting ZIPF's Law

freq_by_rank %>%

  ggplot(aes(rank, `term frequency`, color=news))+
```

```
 geom_abline(intercept=-0.62, slope= -1.1, color='gray50', linetype=2)+
 geom_line(size= 1.1, alpha = 0.8, show.legend = FALSE)+
 scale_x_log10()+
 scale_y_log10()
```

[Output]



[Input]

```
# TF_IDF
all_words <- all_tokens %>%
 bind_tf_idf(word, line, n)

all_words %>%
 arrange(desc(tf_idf))
```

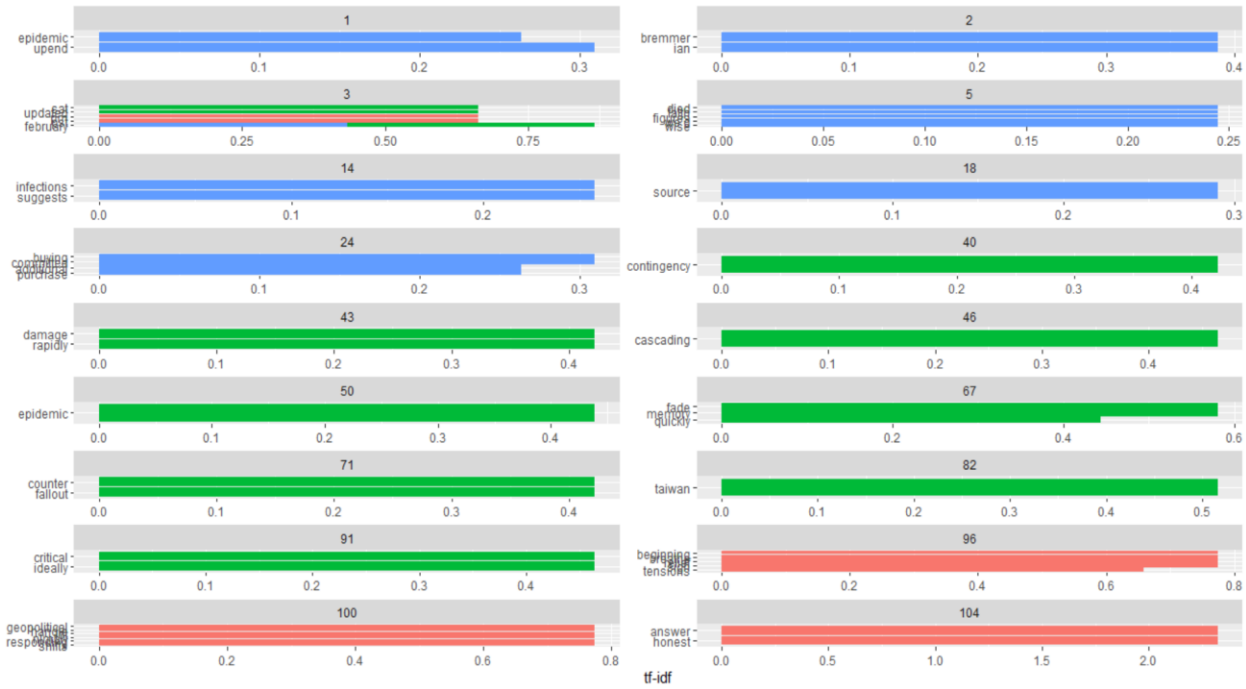
[Output]

```
# A tibble: 1,878 x 7
    line word             n news         tf   idf tf_idf
   <int> <chr>        <int> <chr>     <dbl> <dbl>  <dbl>
 1   104 answer           1 Bloomberg 0.5    4.64  2.32
 2   104 honest           1 Bloomberg 0.5    4.64  2.32
 3    96 beginning        1 Bloomberg 0.167  4.64  0.774
 4    96 breathe          1 Bloomberg 0.167  4.64  0.774
 5    96 relief           1 Bloomberg 0.167  4.64  0.774
 6    96 sigh             1 Bloomberg 0.167  4.64  0.774
 7   100 geopolitical     1 Bloomberg 0.167  4.64  0.774
 8   100 handle           1 Bloomberg 0.167  4.64  0.774
 9   100 lot              1 Bloomberg 0.167  4.64  0.774
10   100 nimbly           1 Bloomberg 0.167  4.64  0.774
# ... with 1,868 more rows
```

[Input]

```
# looking at the graphical apprach:
all_words %>%
 arrange(desc(tf_idf)) %>%
 mutate(word=factor(word, levels=rev(unique(word)))) %>%
 group_by(news) %>%
 top_n(15) %>%
 ungroup %>%
 ggplot(aes(word, tf_idf, fill=news))+
 geom_col(show.legend=FALSE)+
 labs(x=NULL, y="tf-idf")+
 facet_wrap(~line, ncol=2, scales="free")+
 coord_flip()
```

[Output]



[Input]

```
################################################
###### N-grams and tokenizing ################
```

```
###############################################


CNN_bigrams <- CNN_df %>%

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

TIME_bigrams <- TIME_df %>%

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

BLB_bigrams <- BLB_df %>%

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

all_bigrams <- all_df %>%

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


CNN_bigrams%>%

  count(bigram, sort = TRUE)

TIME_bigrams%>%

  count(bigram, sort = TRUE)

BLB_bigrams%>%

  count(bigram, sort = TRUE)

all_bigrams%>%

  count(bigram, sort = TRUE)
```

[Output]

```
> CNN_bigrams%>%
+   count(bigram, sort = TRUE)
# A tibble: 1,234 x 2
   bigram           n
   <chr>        <int>
 1 the virus       12
 2 in the           8
 3 the world        7
 4 if the           6
 5 of the           6
 6 according to     5
 7 virus is         5
 8 and the          4
 9 but the          4
10 in a             4
# ... with 1,224 more rows
> TIME_bigrams%>%
+   count(bigram, sort = TRUE)
# A tibble: 493 x 2
   bigram             n
   <chr>          <int>
```

```
 1 in the             4
 2 of the             4
 3 and the            3
 4 global economy     3
 5 number of          3
 6 to the             3
 7 are now            2
 8 china and          2
 9 for the            2
10 from beijing       2
# ... with 483 more rows
> BLB_bigrams%>%
+   count(bigram, sort = TRUE)
# A tibble: 1,558 x 2
   bigram        n
   <chr>     <int>
 1 the virus    11
 2 in china     10
 3 in the        9
 4 and the       7
 5 of the        7
 6 from the      6
 7 on the        6
 8 to the        6
 9 for the       5
10 more than     5
# ... with 1,548 more rows
> all_bigrams%>%
+   count(bigram, sort = TRUE)
# A tibble: 3,087 x 2
   bigram        n
   <chr>     <int>
 1 the virus    23
 2 in the       21
 3 of the       17
 4 and the      14
 5 in china     13
 6 to the       13
 7 the world    12
 8 from the     10
 9 more than    10
10 in a          8
# ... with 3,077 more rows
```

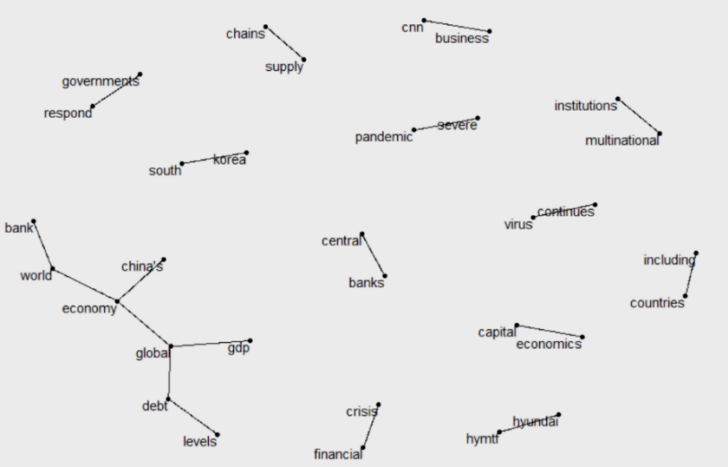
[Input]

```
CNN_bigrams_separated <- CNN_bigrams %>%

  separate(bigram, c("word1", "word2"), sep = " ")

TIME_bigrams_separated <- TIME_bigrams %>%

  separate(bigram, c("word1", "word2"), sep = " ")

BLB_bigrams_separated <- BLB_bigrams %>%

  separate(bigram, c("word1", "word2"), sep = " ")

all_bigrams_separated <- all_bigrams %>%

  separate(bigram, c("word1", "word2"), sep = " ")
```

```
CNN_bigrams_filtered <- CNN_bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
TIME_bigrams_filtered <- TIME_bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
BLB_bigrams_filtered <- BLB_bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
all_bigrams_filtered <- all_bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

#creating the new bigram, "no-stop-words":
CNN_bigram_counts <- CNN_bigrams_filtered %>%
  count(word1, word2, sort = TRUE)
TIME_bigram_counts <- TIME_bigrams_filtered %>%
  count(word1, word2, sort = TRUE)
BLB_bigram_counts <- BLB_bigrams_filtered %>%
  count(word1, word2, sort = TRUE)
all_bigram_counts <- all_bigrams_filtered %>%
  count(word1, word2, sort = TRUE)

# Visualizing a bigram network
CNN_bigram_graph <- CNN_bigram_counts %>%
  filter(n>1) %>%
  graph_from_data_frame()

ggraph(CNN_bigram_graph, layout = "fr") +
```

```
  geom_edge_link()+
  geom_node_point()+
  geom_node_text(aes(label=name), vjust =1, hjust=1)
```

[Output]



[Input]

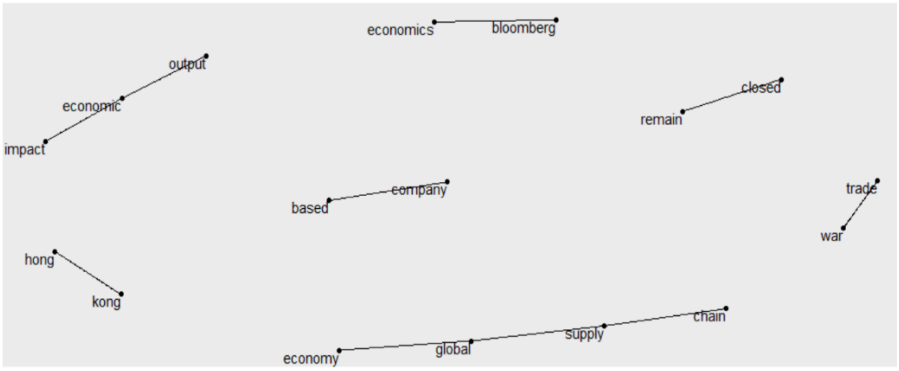
```
TIME_bigram_graph <- TIME_bigram_counts %>%
  filter(n>1) %>%
  graph_from_data_frame()

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

[Output]

[Input]

```
BLB_bigram_graph <- BLB_bigram_counts %>%
  filter(n>1) %>%
  graph_from_data_frame()


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

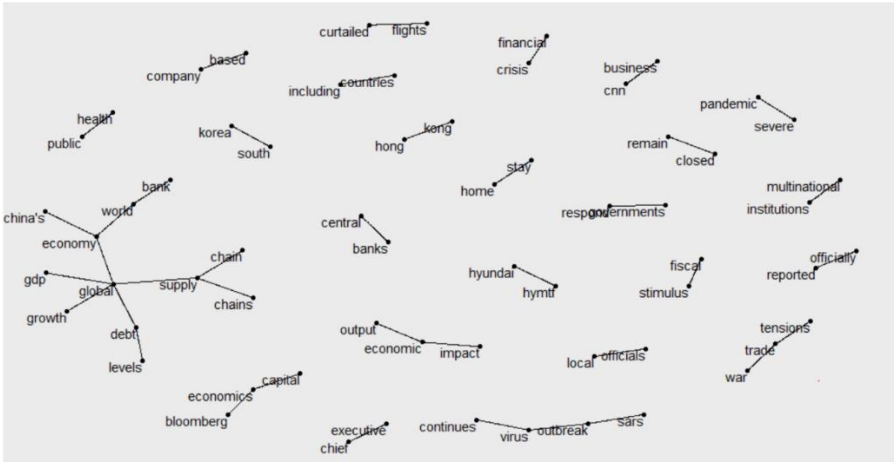
[Output]



[Input]

```
all_bigram_graph <- all_bigram_counts %>%
  filter(n>1) %>%
  graph_from_data_frame()


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

[Output]


curtailed
flights
financial
crisis
based
company
including
countries
business
cnn
pandemic
severe
health
public
korea
south
keng
hong
remain
closed
stay
home
multinational
institutions
bank
world
china's
economy
central
banks
chain
gdp
global
supply
chains
growth
debt
levels
hyundai
hymtl
fiscal
stimulus
officially
reported
output
economic
impact
tensions
trade
war
local
officials
capital
economics
bloomberg
executive
chief
continues
virus
outbreak
sars