Análisis de frecuencias de palabras

HR Analytics: Teoría y Práctica

http://pablohaya.com/contact

01/2022

Introducción

Pregunta: ¿Cómo podemos extraer la temática de un documento?

Una aproximación básica pero bastante efectiva es fijarse en las palabras más frecuentes. El problema es que muchas de estas palabras son frecuentes porque son comunes.

Se pueden utilizar lista de *stop words* para eliminarlas, pero ya hemos visto que dependiendo del texto algunas veces una palabra te interesa quitarla y otras veces no.

Una aproximación clásica que da buenos resultados es **tf-idf** (*term frequency - inverse document frecuency).

Se computa la frecuencia de términos (*tf*) y se ajusta para según la frecuencia de aparición en los distintos documentos (**idf**). De esta manera, la puntuación más alta la obtienen palabras con eleveda frecuencia que aparecen en pocos documentos.

Las palabras frecuentes que aparecen en muchos documentos se ven penalizadas.

Análisis de frecuencia

Leemos de nuevo el *dataset*, y tokenizamos la columna `summary`

```
reviews <- read_csv(here("data/employee_reviews_10000.csv")

tidy_reviews <- reviews %>%
  select(company, summary) %>%
  unnest_tokens(word, summary)

tidy_reviews
```

```
## # A tibble: 43,693 x 2
##    company word
##    <chr>   <chr>
##  1 apple   better
##  2 apple   for
##  3 apple   the
##  4 apple   short
##  5 apple   term
##  6 google  head
##  7 google  of
##  8 google  industry
##  9 amazon  overall
## 10 amazon  is
## # ... with 43,683 more rows
```

Calculamos la frecuencia de cada palabra absoluta (`n`) y el total de las palabras para cada compañía (`total`)

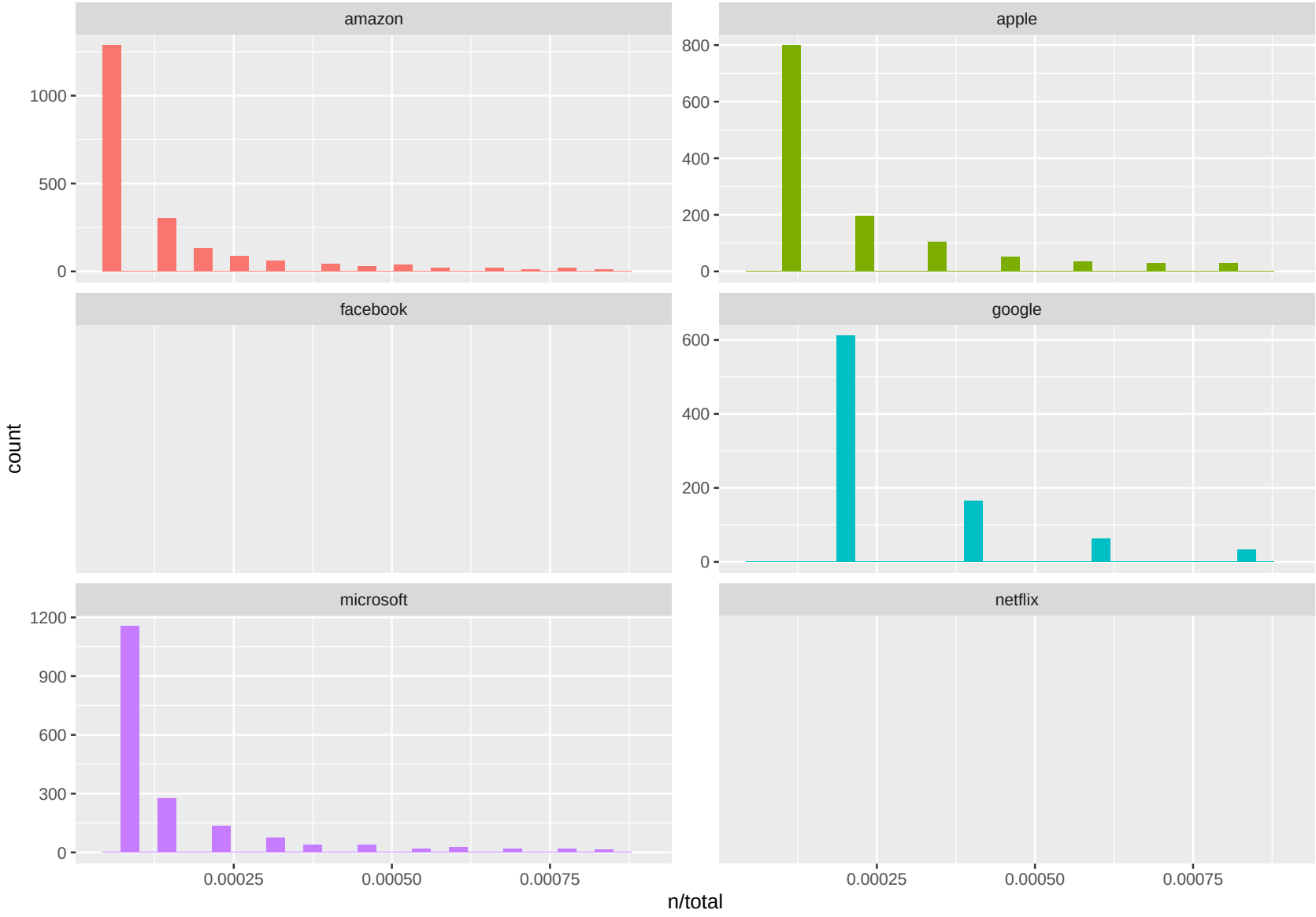
```
reviews_words <- tidy_reviews %>%
  count(company, word, sort = TRUE)

total_words <- reviews_words %>%
  group_by(company) %>%
  summarize(total = sum(n))

reviews_words <- left_join(reviews_words, total_words)

reviews_words
```

```
## Joining, by = "company"
```

```
## # A tibble: 7,304 x 4
##    company   word        n total
##    <chr>     <chr>   <int> <int>
##  1 amazon    work      594 15415
##  2 amazon    to        579 15415
##  3 microsoft great     573 12985
##  4 microsoft to        550 12985
##  5 microsoft work      449 12985
##  6 microsoft company   427 12985
##  7 apple     great     416  8736
##  8 amazon    great     404 15415
##  9 amazon    amazon    382 15415
## 10 amazon    place     370 15415
## # ... with 7,294 more rows
```

Visualizamos los histogramas de frecuencia de palabras para cada compañia:

```
ggplot(reviews_words, aes(n/total, fill = company)) +
  geom_histogram(show.legend = FALSE) +
  xlim(NA, 0.0009) +
  facet_wrap(~company, ncol = 2, scales = "free_y")
```


amazon
apple
facebook
google
microsoft
netflix
count
n/total
0
500
1000
200
400
600
800
300
900
1200
0.00025
0.00050
0.00075

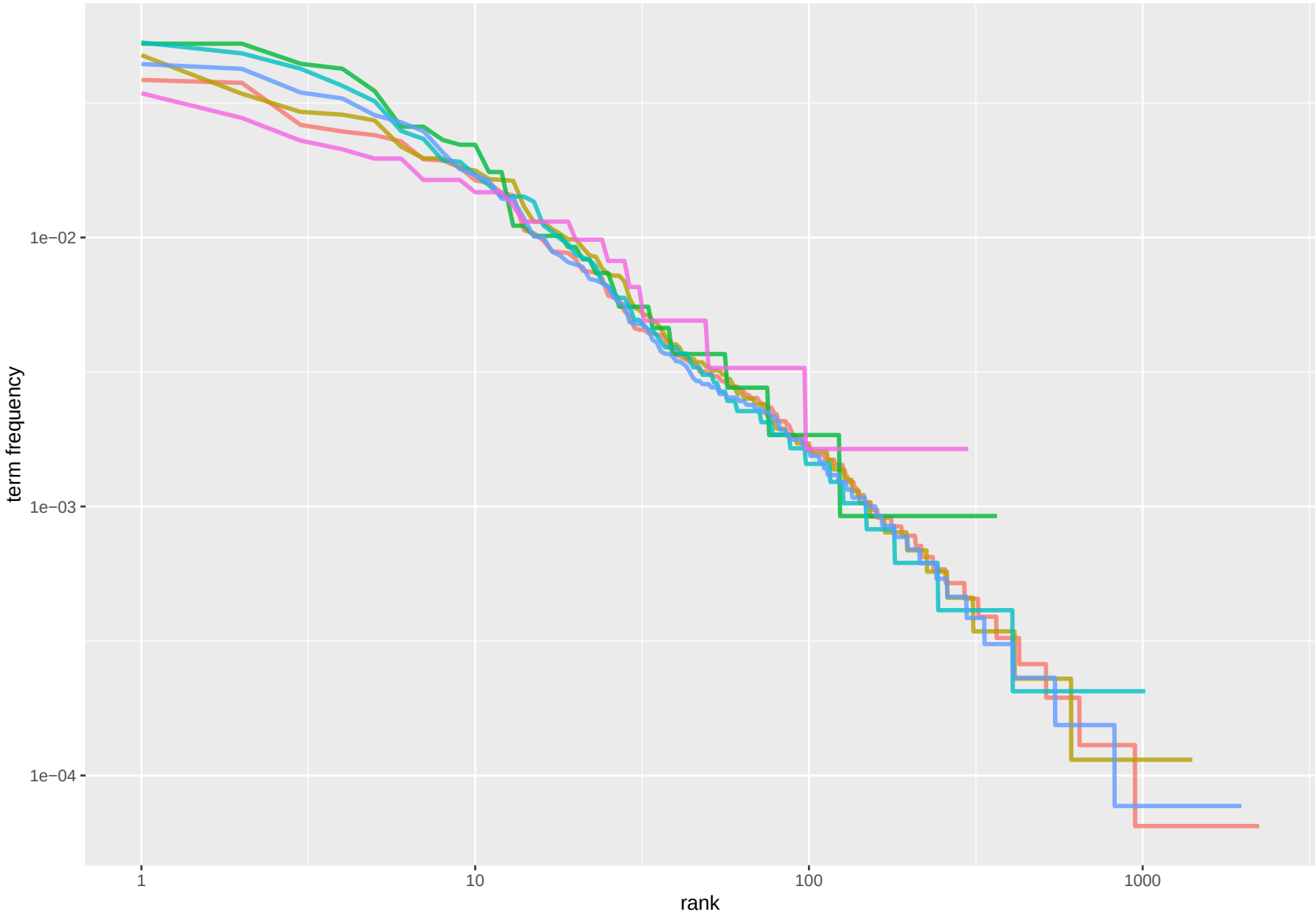
Ley de Zipf

La lye de Zipf establece que la frecuencia de aparición de una palabra es inversamente propocional a su orden en la lista de frecuencias

```
freq_by_rank <- reviews_words %>%
  group_by(company) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total) %>%
  ungroup()

freq_by_rank
```

```
## # A tibble: 7,304 x 6
##    company   word        n total  rank `term frequency`
##    <chr>     <chr>   <int> <int> <int>            <dbl>
##  1 amazon    work      594 15415     1           0.0385
##  2 amazon    to        579 15415     2           0.0376
##  3 microsoft great     573 12985     1           0.0441
##  4 microsoft to        550 12985     2           0.0424
##  5 microsoft work      449 12985     3           0.0346
##  6 microsoft company   427 12985     4           0.0329
##  7 apple     great     416  8736     1           0.0476
##  8 amazon    great     404 15415     3           0.0262
##  9 amazon    amazon    382 15415     4           0.0248
## 10 amazon    place     370 15415     5           0.0240
## # ... with 7,294 more rows
```

Visualizamos el resultado



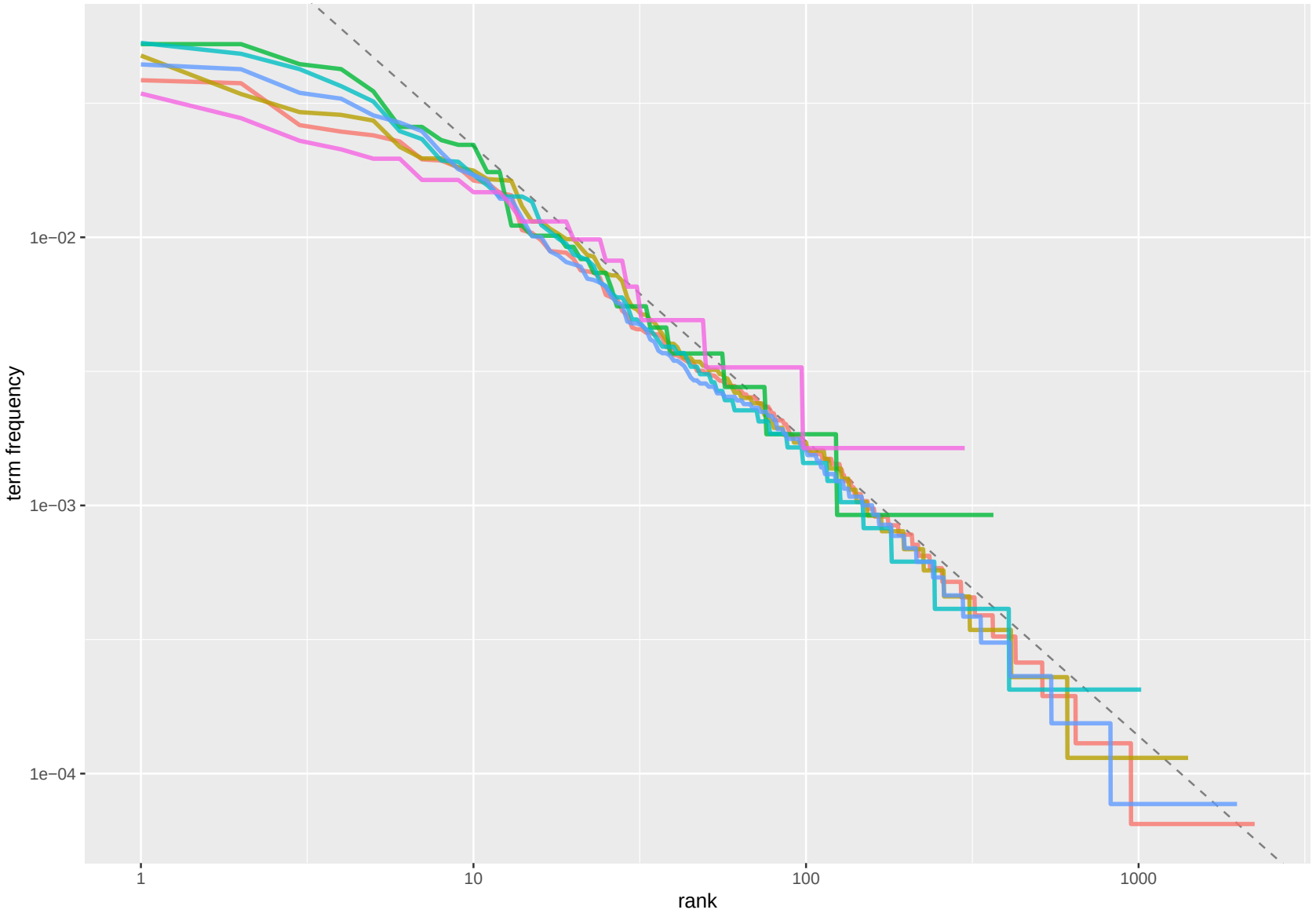
Podemos calcular el resultado teórico, y contrastar como se aproxima al resultado experimental

```
rank_subset <- freq_by_rank %>%
  filter(rank < 500,
         rank > 10)

lm(log10(`term frequency`) ~ log10(rank),
                             data = rank_subset)
```

```
##
## Call:
## lm(formula = log10(`term frequency`) ~ log10(rank), data
##
## Coefficients:
## (Intercept)  log10(rank)
##     -0.5604      -1.1109
```

La línea de puntos representa como debería variar la frecuencia de palabras si se cumpliera perfectamente la ley de Zipf

```
freq_by_rank %>%
  ggplot(aes(rank, `term frequency`, color = company)) +
  geom_abline(intercept = -0.56, slope = -1.1,
              color = "gray50", linetype = 2) +
  geom_line(size = 1.1, alpha = 0.8,
            show.legend = FALSE) +
  scale_x_log10() +
  scale_y_log10()
```


term frequency
1e−02
1e−03
1e−04
1
10
100
1000
rank

Ejemplo práctico

Entendido como se distribuyen las palabras por frecuencia, vamos a ver con con **tf-idf** somos capaces de ajustar esa frecuencia para cada documento. De esta manera reducimos la importancia de palabras frecuentes que aparecen en muchos documentos.

Quitamos `facebook` y `netflix` dado que tienen muy pocas revisiones comparadas con el resto.

```
reviews_tf_idf <- reviews_words %>%
  filter(company != 'facebook', company != 'netflix') %>%
  bind_tf_idf(word, company, n)

reviews_tf_idf
```

```
## # A tibble: 6,638 x 7
##    company   word        n total     tf   idf tf_idf
##    <chr>     <chr>   <int> <int>  <dbl> <dbl>  <dbl>
## 1 amazon    work      594 15415 0.0385     0      0
## 2 amazon    to        579 15415 0.0376     0      0
## 3 microsoft great     573 12985 0.0441     0      0
## 4 microsoft to        550 12985 0.0424     0      0
## 5 microsoft work      449 12985 0.0346     0      0
## 6 microsoft company   427 12985 0.0329     0      0
## # ... with 6,632 more rows
```

El valor de **idf** y, en consecuencia, *tf-idf** es cero en palabras que son extremadamente comunes. Esto es así porque son palabras que aparecen en las revisiones de todas las compañias. Dependiendo de en cuantas compañías aparece esa palabras tendrá un valor distinto.

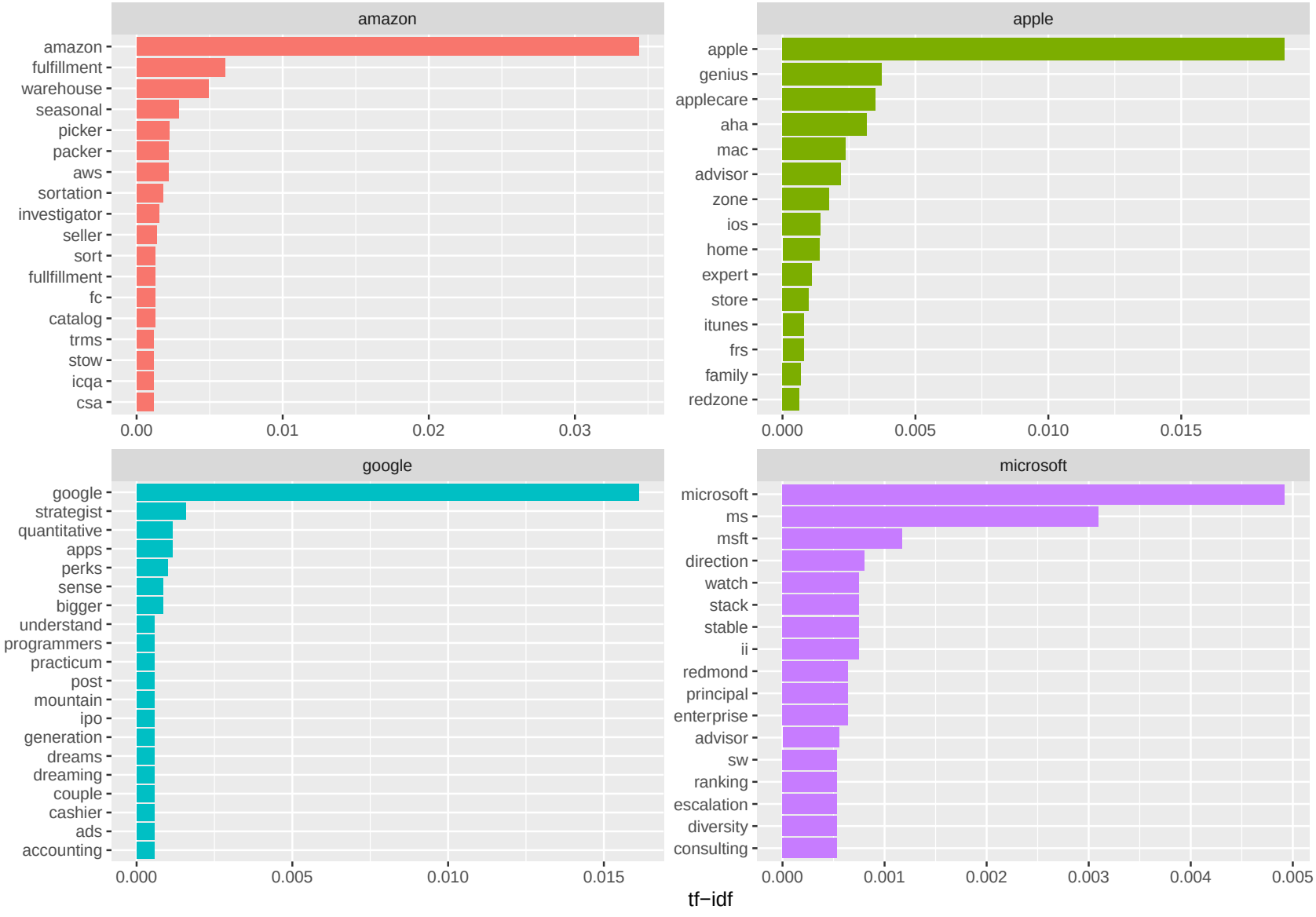
El valor máximo sería que aparecería en una sola compañia.

```
reviews_tf_idf %>%
  select(-total) %>%
  arrange(desc(tf_idf))
```

```
## # A tibble: 6,638 x 6
##    company   word            n      tf   idf  tf_idf
##    <chr>     <chr>       <int>   <dbl> <dbl>   <dbl>
##  1 amazon    amazon        382 0.0248  1.39  0.0344
##  2 apple     apple         238 0.0272  0.693 0.0189
##  3 google    google        113 0.0232  0.693 0.0161
##  4 amazon    fulfillment    67 0.00435 1.39  0.00603
##  5 microsoft microsoft     222 0.0171  0.288 0.00492
##  6 amazon    warehouse     109 0.00707 0.693 0.00490
##  7 apple     genius         47 0.00538 0.693 0.00373
##  8 apple     applecare      22 0.00252 1.39  0.00349
##  9 apple     aha            20 0.00229 1.39  0.00317
## 10 microsoft ms             29 0.00223 1.39  0.00310
## # ... with 6,628 more rows
```

Visualizamos las palabras para cada compañia que tiene mayor valor **tf-idf** y que representan las temáticas que son características revisiones de cada compañia.

```
reviews_tf_idf %>%
  group_by(company) %>%
  slice_max(tf_idf, n = 15) %>%
  ungroup() %>%
  ggplot(aes(tf_idf, reorder_within(word, tf_idf, company)
  geom_col(show.legend = FALSE) +
  facet_wrap(~company, ncol = 2, scales = "free") +
  scale_y_reordered() +
  labs(x = "tf-idf", y = NULL)
```


amazon
amazon
fulfillment
warehouse
seasonal
picker
packer
aws
sortation
investigator
seller
sort
fullfillment
fc
catalog
trms
stow
icqa
csa
0.00
0.01
0.02
0.03
apple
apple
genius
applecare
aha
mac
advisor
zone
ios
home
expert
store
itunes
frs
family
redzone
0.000
0.005
0.010
0.015
google
google
strategist
quantitative
apps
perks
sense
bigger
understand
programmers
practicum
post
mountain
ipo
generation
dreams
dreaming
couple
cashier
ads
accounting
0.000
0.005
0.010
0.015
microsoft
microsoft
ms
msft
direction
watch
stack
stable
ii
redmond
principal
enterprise
advisor
sw
ranking
escalation
diversity
consulting
0.000
0.001
0.002
0.003
0.004
0.005
tf−idf

A practicar

Ejercicio. Repetir los análisis con las columnas `pros` y `cons`.