Assignment: Examining Employee Organizational Reviews with Text Analytics

Emma Kruis

2020-01-25

Instructions

This assignment reviews the *Text Analytics* content. You will use the *text_analytics.Rmd* file I reviewed as part of the lectures for this week to complete this assignment. You will *copy and paste* relevant code from that file and update it to answer the questions in this assignment. You will respond to questions in each section after executing relevant code to answer a question. You will submit this assignment to its *Submissions* folder on *D2L*. You will submit *two* files:

1. this completed *R Markdown* script, and
2. as a first preference, a *PDF* (if you already installed `TinyTeX` properly), as a second preference, a *Microsfot Word* (if your computer has *Microsoft Word*) document, or, as a third preference, an *HTML* (if you did *not* install `TinyTeX` properly and your computer does *not* have *Microsoft Word*) file to *D2L*.

To start:

First, create a folder on your computer to save all relevant files for this course. If you did not do so already, you will want to create a folder named *mgt_592* that contains all of the materials for this course.

Second, inside of *mgt_592*, you will create a folder to host assignments. You can name that folder *assignments*.

Third, inside of *assignments*, you will create folders for each assignment. You can name the folder for this first assignment: *text_analytics*.

Fourth, create three additional folders in *text_analytics* named *scripts*, *data*, and *plots*. Store this script in the *scripts* folder and the data for this assignment in the *data* folder.

Fifth, go to the *File* menu in *RStudio*, select *New Project...*, choose *Existing Directory*, go to your *~/mgt_592/assignments/text_analytics* folder to select it as the top-level directory for this **R Project**.

Global Settings

The first code chunk sets the global settings for the remaining code chunks in the document. Do *not* change anything in this code chunk.

Load Packages

In this code chunk, we load the following packages:

1. **here**,

2. **tidyverse**,
3. **tidygraph**,
4. **ggraph**,
5. **tidytext**,
6. **ggwordcloud**,
7. **widyr**, and
8. **topicmodels**.

Make sure you installed these packages when you reviewed the analytical lecture.

We will use functions from these packages to examine the data. Do *not* change anything in this code chunk.

```
### load libraries for use in current working session
## here for project work flow
library(here)

## tidyverse for data manipulation and plotting
# loads eight different libraries simultaneously
library(tidyverse)

## tidygraph for network data
library(tidygraph)

## ggraph to plot networks
library(ggraph)

## tidytext for text analytics
library(tidytext)

## ggwordcloud for word clouds
library(ggwordcloud)

## widyr for tidy data processing
library(widyr)

## topicmodels for latent Dirichlet allocation
library(topicmodels)
```

Task 1: Import Data

We will use the same data as in the analytical lecture: **amazon_reviews.txt** and **google_reviews.txt**. After you load the data, then you will execute other commands on the data.

Task 1.1

Use the **read_delim()** and **here()** functions to load the data files for this working session. Save the data as the objects **amazon_raw** and **google_raw**. Use **glimpse()** to preview both data tables.

Questions 1.1: Answer these questions: (1) How many *observations* are there in the **amazon_raw** data table? (2) What are the *first four words* of the *first comment of the cons* in the **google_raw** data table?

Responses 1.1: *(1) 500 obs (2) It is becoming larger.*

```r
##important data objects
amazon_raw <- read_delim(
  here("data", "amazon_reviews.txt"),
  # delimiter
  delim = "|"
)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   pg_num = col_double(),
##   url = col_character(),
##   pros = col_character(),
##   cons = col_character()
## )
```

```r
#preview data
glimpse(amazon_raw)
```

```
## Rows: 500
## Columns: 4
## $ pg_num <dbl> 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 49, 49, 49, 49, 49, 49,~
## $ url    <chr> "https://www.glassdoor.com/Reviews/Amazon-com-Reviews-E6036_P50~
## $ pros   <chr> "You're surrounded by smart people and the projects are interes~
## $ cons   <chr> "Internal tools proliferation has created a mess for trying to ~
```

```r
##import data objects
google_raw <- read_delim(
  here("data", "google_reviews.txt"),
  delim = "|"
)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   pg_num = col_double(),
##   url = col_character(),
##   pros = col_character(),
##   cons = col_character()
## )
```

```r
## preview data
glimpse(google_raw)
```

```
## Rows: 501
## Columns: 4
## $ pg_num <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, ~
## $ url    <chr> "https://www.glassdoor.com/Reviews/Google-Reviews-E9079_P1.htm"~
## $ pros   <chr> "* If you're a software engineer, you're among the kings of the~
## $ cons   <chr> "* It *is* becoming larger, and with it comes growing pains: bu~
```

Task 2: Clean Data

For this task, you will clean the data.

Task 2.1

Create a new data table named **emp_reviews**. To create it, *row bind* **amazon_raw** and **google_raw** and set **.id** to **org**. Add an **id** variable to identify the rows of the new data table, change **org** to a *factor*, and recode the levels of **org** to identify **amazon** and **google** rows. Select the **id**, **org**, **pros**, and **cons** columns. Group the data table *row-wise*. Filter the data table to include only rows with at least one non-missing value for the **pros** and **cons** columns. Remove the *row-wise* groups.

Apply **glimpse()** to **emp_reviews** to preview the data table.

Questions 2.1: Answer these questions: (1) How many *observations* are there in the **emp_reviews** data table? (2) What are the *first four words* of the *first comment* in the **pros** column?

Responses 2.1: *(1) 998 (2) Interal tools prolifeation has.*

```
##convert variables
emp_reviews <- amazon_raw %>%
  bind_rows(
    google_raw,
    .id = "org"
  ) %>%
  mutate(
    id = row_number(),
    org = as_factor(org),
    org = fct_recode(
      org,
      "amazon" = "1",
      "google" = "2"
    )
  ) %>%
  select(id, org, pros, cons) %>%
  rowwise() %>%
  filter(
    any(
      !is.na(
        c_across(pros:cons)
      )
    )
  ) %>%
  ungroup()


## glimpse data to confirm changes
glimpse(emp_reviews)
```

```
## Rows: 998
## Columns: 4
## $ id   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19~
## $ org  <fct> amazon, amazon, amazon, amazon, amazon, amazon, amazon, amazon, a~
## $ pros <chr> "You're surrounded by smart people and the projects are interesti~
## $ cons <chr> "Internal tools proliferation has created a mess for trying to ge~
```

Task 2.2

Overwrite **emp_reviews** by making it a long data table. Pivot the **pros** and **cons** columns, identify the pivoted columns in a column named **type**, and identify the values of the pivoted columns in a column named **comment**. Filter for rows with non-missing *comments*.

Apply **glimpse()** to **emp_reviews** to preview the data table.

Questions 2.2: Answer these questions: (1) How many *observations* are there in the updated **emp_reviews** data table? (2) What are the *first four words* of the *third row* in the **comment** column?

Responses 2.2: *(1) 1994 (2) Brand name is great.*

```
### make long data table
## overwrite working data
emp_reviews <- emp_reviews %>%
  pivot_longer(
    cols = c(pros, cons),
    names_to = "type",
    values_to = "comment"
  ) %>%
  filter(
    !is.na(comment)
  )

## preview
glimpse(emp_reviews)
```

```
## Rows: 1,994
## Columns: 4
## $ id      <int> 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10, ~
## $ org     <fct> amazon, amazon, amazon, amazon, amazon, amazon, amazon, amazon~
## $ type    <chr> "pros", "cons", "pros", "cons", "pros", "cons", "pros", "cons"~
## $ comment <chr> "You're surrounded by smart people and the projects are intere~
```

Task 3: Tokenize Text

For this task, you will tokenize the comments as unigrams.

Task 3.1

Create a new data table named **unigrams**. Pipe **emp_reviews** into **unnest_tokens()**. Tokenize the **comments** column as unigrams, setting the name of the token column to **word**. Count the **word** column in **unigrams** while *arranging* the result by *descending* count.

Questions 3.1: Answer these questions: (1) How many *distinct* words are there in the **unigrams** data table? (2) What is the *count* of the most frequent word?

Responses 3.1: *(1) 3760 (2) 1247.*

```
### unigram comments
unigrams <- emp_reviews %>%
  unnest_tokens(
```

```
    word,
    comment
  )

### count words
unigrams %>%
  count(word) %>%
  arrange(desc(n))
```

```
## # A tibble: 3,770 x 2
##    word      n
##    <chr> <int>
##  1 to     1247
##  2 the    1174
##  3 and    1163
##  4 a       841
##  5 of      764
##  6 work    656
##  7 is      645
##  8 you     592
##  9 great   406
## 10 for     403
## # ... with 3,760 more rows
```

Task 3.2

Overwrite **unigrams** by performing an **anti-join** with **stop_words**. Pipe **unigrams** into **anti_join()**. Specify **stop_words** as an input and set the key to **word** in **anti_join()**. Count the **word** column in **unigrams** while *arranging* the result by *descending* count.

Questions 3.2: Answer these questions: (1) How many *distinct* words are there in the updated **unigrams** data table? (2) What is the *count* of the most frequent word?

Responses 3.2: *(1) 3231 (2) 371* .

```
### remove stop words
unigrams <- unigrams %>%
  anti_join(
    stop_words,
    by = "word"
  )

### count words
unigrams %>%
  count(word) %>%
  arrange(desc(n))
```

```
## # A tibble: 3,231 x 2
##    word            n
##    <chr>       <int>
##  1 people        371
##  2 company       301
##  3 time          191
```

```
## 4 benefits      189
## 5 lot           171
## 6 hours         160
## 7 pay           154
## 8 google        145
## 9 management    128
## 10 environment  121
## # ... with 3,221 more rows
```

Task 4: Term Frequency - Inverse Document Frequency

For this task, you will calculate the term and inverse document frequencies.

Task 4.1

Create a data table named **doc_count** from **unigrams**. This data table should consist of the *distinct* combinations of **id**, **org**, and **type**. Count the number of combinations of **org** and **type**. Name the count column **n_doc** and sort the data table by descending count.

Create a data table named **doc_word_count** from **unigrams**. This data table should consist of the *distinct* combinations of **id**, **org**, **type**, and **word**. Count the number of combinations of **org**, **type**, and **word**. Name the count column **n_doc_word** and sort the data table by descending count.

Create a data table named **word_count** from **unigrams**. Count the number of combinations of **org**, **type**, and **word**. Name the count column **n_word** and sort the data table by descending count.

Overwrite **word_count** with two *left joins*. Pipe **word_count** into **left_join()** with **doc_word_count** joining by **org**, **type**, and **word**. Pipe the result into **left_join()** with **doc_count** joining by **org** and **type**.

Overwrite **word_count** by calculating new variables from groups. Pipe **word_count** into groups formed by **org** and **type**. Calculate the *term frequency* (named **tf**), *document frequency* (**df**), *log of inverse document frequency* (**idf**), and *term frequency - inverse document frequency* (**tf_idf**) with the appropriate formulas. Remove the groups. Prnt a preview of the updated **word_count**.

Questions 4.1: Answer these questions: (1) What is the *absolute frequency* of the *first* listed word? (2) How many *documents* contain the *second* listed word? (3) In how many *documents* could have the *third* listed word appeared given the word's *organization* and *type* identifier? (4) What is the *term frequency - inverse document frequency* of the *fourth* listed word?

Responses 4.1: *(1) 0.0445 (2) 94 (3) 492 (4) 0.0460.*

```
### number of distinct comments per organization and type
doc_count <- unigrams %>%
  distinct(id, org, type) %>%
  count(org, type, name = "n_doc", sort = TRUE)


### number of comments containing each word
doc_word_count <- unigrams %>%
  distinct(id, org, type, word) %>%
  count(org, type, word, name = "n_doc_word", sort = TRUE)


### number of words per organization and type
```

```r
word_count <- unigrams %>%
  count(org, type, word, name = "n_word", sort = TRUE)

### join count data tables
word_count <- word_count %>%
  left_join(
    doc_word_count,
    by = c("org", "type", "word")
  ) %>%
  left_join(
    doc_count,
    by = c("org", "type")
  )

### compute frequency statistics
word_count <- word_count %>%
  group_by(org, type) %>%
  mutate(
    tf = n_word / sum(n_word),
    df = n_doc_word / n_doc,
    idf = log(n_doc / n_doc_word),
    tf_idf = tf * idf
  ) %>%
  ungroup()

## preview
word_count
```

```
## # A tibble: 5,188 x 10
##    org    type  word     n_word n_doc_word n_doc     tf    df   idf tf_idf
##    <fct>  <chr> <chr>     <int>      <int> <int>  <dbl> <dbl> <dbl>  <dbl>
##  1 google pros  people      149        136   495 0.0445 0.275  1.29 0.0575
##  2 google cons  company     106         94   486 0.0316 0.193  1.64 0.0518
##  3 amazon pros  pay          92         89   492 0.0243 0.181  1.71 0.0415
##  4 google pros  perks        91         91   495 0.0272 0.184  1.69 0.0460
##  5 google pros  food         90         87   495 0.0269 0.176  1.74 0.0467
##  6 amazon pros  people       88         81   492 0.0232 0.165  1.80 0.0419
##  7 google pros  benefits     87         85   495 0.0260 0.172  1.76 0.0458
##  8 amazon pros  benefits     86         84   492 0.0227 0.171  1.77 0.0401
##  9 google pros  google       85         75   495 0.0254 0.152  1.89 0.0479
## 10 amazon pros  company      83         75   492 0.0219 0.152  1.88 0.0412
## # ... with 5,178 more rows
```

Task 5: Examine Words

For this task, you will examine the unigrams.

Task 5.1

Create a data table named **top_word_count**. Pipe **word_count** into groups formed by **org** and **type**. Slice for the *top 15* words by *term frequency - inverse document frequency* removing any ties. Remove groups. Change to *title case* **org**, **type**, and **word**. Calculate a new variable named **word_id** reordering **word** within **org** and **type** by *term frequency - inverse document frequency*. Print all rows of the data table.

Questions 5.1: Answer these questions: (1) For *Amazon cons*, what is the top word by *term frequency - inverse document frequency*?
(2) For *Google pros*, what is the top word by *term frequency - inverse document frequency*?

Responses 5.1: *(1) 0.0419 (2) 0.0575.*

```
## compare top words by organization, comment type
top_word_count <- word_count %>%
  group_by(org, type) %>%
  slice_max(
    tf_idf,
    n = 15,
    with_ties = FALSE
  ) %>%
  ungroup() %>%
  mutate(
    across(
      .cols = c(org, type, word),
      .fns = str_to_title
    ),
    word_id = reorder_within(
      word,
      tf_idf,
      list(org, type)
    )
  )

## print
top_word_count %>%
  print(n = Inf)
```

```
## # A tibble: 60 x 11
##    org   type  word  n_word n_doc_word n_doc      tf     df   idf tf_idf word_id
##    <chr> <chr> <chr>  <int>      <int> <int>   <dbl>  <dbl> <dbl>  <dbl> <fct>
##  1 Amaz~ Cons  Hours     78         67   485 0.0192  0.138   1.98 0.0381 Hours_~
##  2 Amaz~ Cons  Peop~     70         63   485 0.0173  0.130   2.04 0.0353 People~
##  3 Amaz~ Cons  Mana~     61         53   485 0.0151  0.109   2.21 0.0333 Manage~
##  4 Amaz~ Cons  Time      60         54   485 0.0148  0.111   2.20 0.0325 Time__~
##  5 Amaz~ Cons  Life      57         52   485 0.0141  0.107   2.23 0.0314 Life__~
##  6 Amaz~ Cons  Comp~     46         40   485 0.0113  0.0825  2.50 0.0283 Compan~
##  7 Amaz~ Cons  Bala~     46         43   485 0.0113  0.0887  2.42 0.0275 Balanc~
##  8 Amaz~ Cons  Empl~     39         32   485 0.00962 0.0660  2.72 0.0262 Employ~
##  9 Amaz~ Cons  Job       39         36   485 0.00962 0.0742  2.60 0.0250 Job___~
## 10 Amaz~ Cons  Lot       39         38   485 0.00962 0.0784  2.55 0.0245 Lot___~
## 11 Amaz~ Cons  Mana~     35         31   485 0.00864 0.0639  2.75 0.0237 Manage~
## 12 Amaz~ Cons  Day       31         22   485 0.00765 0.0454  3.09 0.0237 Day___~
## 13 Amaz~ Cons  Amaz~     34         29   485 0.00839 0.0598  2.82 0.0236 Amazon~
## 14 Amaz~ Cons  Pay       34         32   485 0.00839 0.0660  2.72 0.0228 Pay___~
## 15 Amaz~ Cons  Hard      33         31   485 0.00814 0.0639  2.75 0.0224 Hard__~
## 16 Amaz~ Pros  Peop~     88         81   492 0.0232  0.165   1.80 0.0419 People~
## 17 Amaz~ Pros  Pay       92         89   492 0.0243  0.181   1.71 0.0415 Pay___~
## 18 Amaz~ Pros  Time      71         54   492 0.0187  0.110   2.21 0.0414 Time__~
```

```
## 19 Amaz~ Pros  Amaz~     78          66    492 0.0206  0.134   2.01 0.0413 Amazon~
## 20 Amaz~ Pros  Comp~     83          75    492 0.0219  0.152   1.88 0.0412 Compan~
## 21 Amaz~ Pros  Bene~     86          84    492 0.0227  0.171   1.77 0.0401 Benefi~
## 22 Amaz~ Pros  Lot       61          53    492 0.0161  0.108   2.23 0.0358 Lot___~
## 23 Amaz~ Pros  Hours     47          45    492 0.0124  0.0915  2.39 0.0296 Hours_~
## 24 Amaz~ Pros  Job       44          39    492 0.0116  0.0793  2.53 0.0294 Job___~
## 25 Amaz~ Pros  Learn     40          37    492 0.0105  0.0752  2.59 0.0273 Learn_~
## 26 Amaz~ Pros  Oppo~     37          32    492 0.00976 0.0650  2.73 0.0267 Opport~
## 27 Amaz~ Pros  Envi~     40          40    492 0.0105  0.0813  2.51 0.0265 Enviro~
## 28 Amaz~ Pros  Oppo~     36          35    492 0.00949 0.0711  2.64 0.0251 Opport~
## 29 Amaz~ Pros  Nice      32          31    492 0.00844 0.0630  2.76 0.0233 Nice__~
## 30 Amaz~ Pros  Fast      31          30    492 0.00818 0.0610  2.80 0.0229 Fast__~
## 31 Goog~ Cons  Comp~    106          94    486 0.0316  0.193   1.64 0.0518 Compan~
## 32 Goog~ Cons  Peop~     64          51    486 0.0191  0.105   2.25 0.0430 People~
## 33 Goog~ Cons  Goog~     60          50    486 0.0179  0.103   2.27 0.0406 Google~
## 34 Goog~ Cons  Hard      48          47    486 0.0143  0.0967  2.34 0.0334 Hard__~
## 35 Goog~ Cons  Mana~     40          37    486 0.0119  0.0761  2.58 0.0307 Manage~
## 36 Goog~ Cons  Time      37          33    486 0.0110  0.0679  2.69 0.0296 Time__~
## 37 Goog~ Cons  Lot       34          32    486 0.0101  0.0658  2.72 0.0275 Lot___~
## 38 Goog~ Cons  Team      28          23    486 0.00834 0.0473  3.05 0.0254 Team__~
## 39 Goog~ Cons  Proj~     26          25    486 0.00774 0.0514  2.97 0.0230 Projec~
## 40 Goog~ Cons  Hours     24          20    486 0.00714 0.0412  3.19 0.0228 Hours_~
## 41 Goog~ Cons  Cons      24          23    486 0.00714 0.0473  3.05 0.0218 Cons__~
## 42 Goog~ Cons  Diff~     24          23    486 0.00714 0.0473  3.05 0.0218 Diffic~
## 43 Goog~ Cons  Life      23          22    486 0.00685 0.0453  3.10 0.0212 Life__~
## 44 Goog~ Cons  Bure~     22          21    486 0.00655 0.0432  3.14 0.0206 Bureau~
## 45 Goog~ Cons  Job       21          19    486 0.00625 0.0391  3.24 0.0203 Job___~
## 46 Goog~ Pros  Peop~    149         136    495 0.0445  0.275   1.29 0.0575 People~
## 47 Goog~ Pros  Goog~     85          75    495 0.0254  0.152   1.89 0.0479 Google~
## 48 Goog~ Pros  Food      90          87    495 0.0269  0.176   1.74 0.0467 Food__~
## 49 Goog~ Pros  Perks     91          91    495 0.0272  0.184   1.69 0.0460 Perks_~
## 50 Goog~ Pros  Bene~     87          85    495 0.0260  0.172   1.76 0.0458 Benefi~
## 51 Goog~ Pros  Free      70          59    495 0.0209  0.119   2.13 0.0444 Free__~
## 52 Goog~ Pros  Smart     79          78    495 0.0236  0.158   1.85 0.0436 Smart_~
## 53 Goog~ Pros  Comp~     66          60    495 0.0197  0.121   2.11 0.0416 Compan~
## 54 Goog~ Pros  Amaz~     59          54    495 0.0176  0.109   2.22 0.0390 Amazin~
## 55 Goog~ Pros  Cult~     59          56    495 0.0176  0.113   2.18 0.0384 Cultur~
## 56 Goog~ Pros  Envi~     53          51    495 0.0158  0.103   2.27 0.0360 Enviro~
## 57 Goog~ Pros  Lots      42          40    495 0.0125  0.0808  2.52 0.0315 Lots__~
## 58 Goog~ Pros  Fun       39          39    495 0.0116  0.0788  2.54 0.0296 Fun___~
## 59 Goog~ Pros  Lot       37          35    495 0.0110  0.0707  2.65 0.0293 Lot___~
## 60 Goog~ Pros  Proj~     31          31    495 0.00925 0.0626  2.77 0.0256 Projec~
```

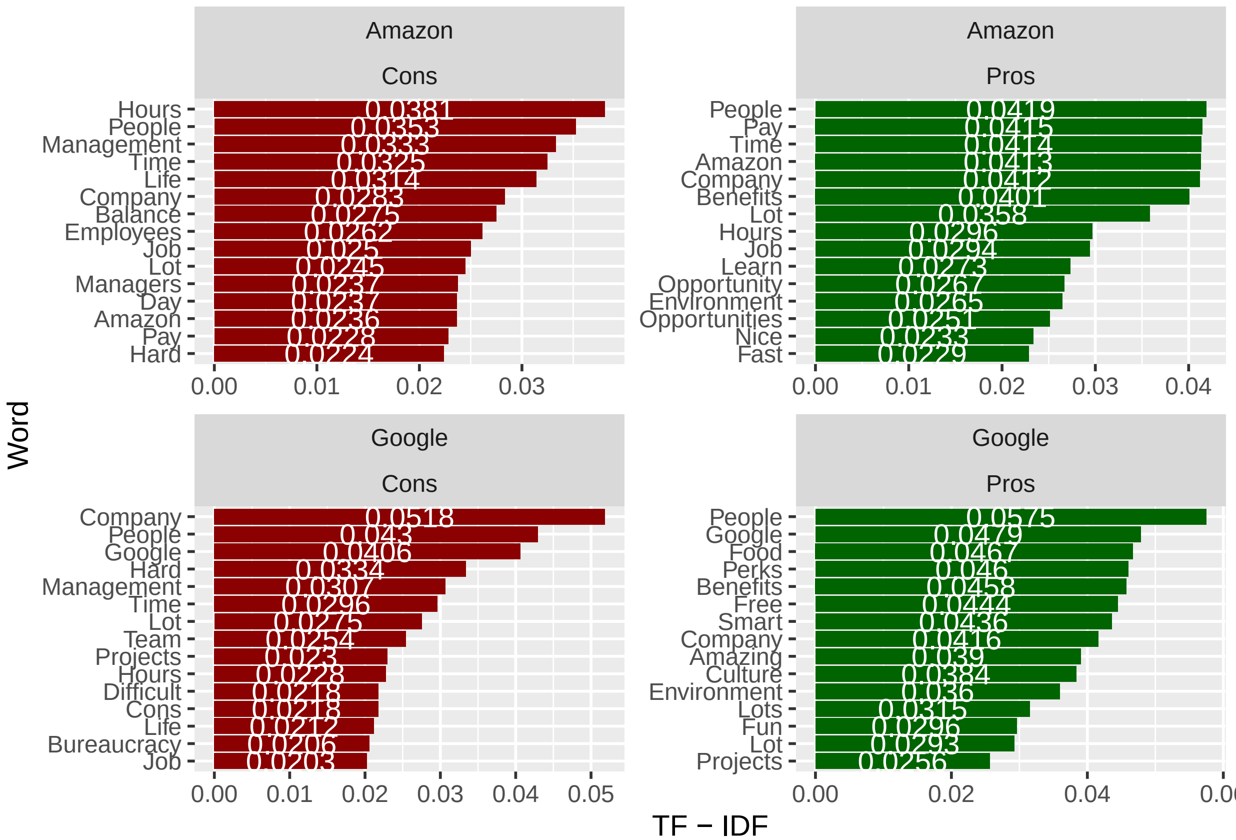
Task 5.2

Create a plot to visualize the top word counts. Call **ggplot()** and set the *data* to **top_word_count**, the *x-axis* to **word_id**, the *y-axis* to **tf_idf**, and the *fill* to **type**. Add a **geom_col()** layer and set the *show legend* option to **FALSE**. Add a **geom_text()** layer and map to *label* rounded values of **tf_idf** to *four* digits, position the values in the middle of the bars, and color the values *white*. Scale the *x-axis* with **scale_x_reordered()**. Create facets of **org** and **type** with **facet_wrap()**. Fill the *cons* bars *dark red* and *pros* bars *dark green*. Flip the coordinates with **coord_flip()**. Label the axes appropriately.

Questions 5.2: Answer these questions: (1) Is the **tf_idf** for **Time** greater for *Amazon cons* or *Amazon pros*? (2) For *Google cons*, what is the top word by *term frequency - inverse document frequency*?

Responses 5.2: *(1) greater for Amazon pros (2) Company.*

```r
### visualize top word counts
ggplot(
  top_word_count,
  aes(
    x = word_id,
    y = tf_idf,
    fill = type
  )
) +
  geom_col(show.legend = FALSE) +
  geom_text(
    aes(
      label = round(
        tf_idf,
        digits = 4
      )
    ),
    position = position_stack(vjust = 0.5),
    color = "white"
  ) +
  scale_x_reordered() +
  facet_wrap(
    vars(org, type),
    scales = "free"
  ) +
  scale_fill_manual(
    values = c(
      "darkred",
      "darkgreen"
    )
  ) +
  coord_flip() +
  labs(x = "Word", y = "TF - IDF")
```



Task 5.3

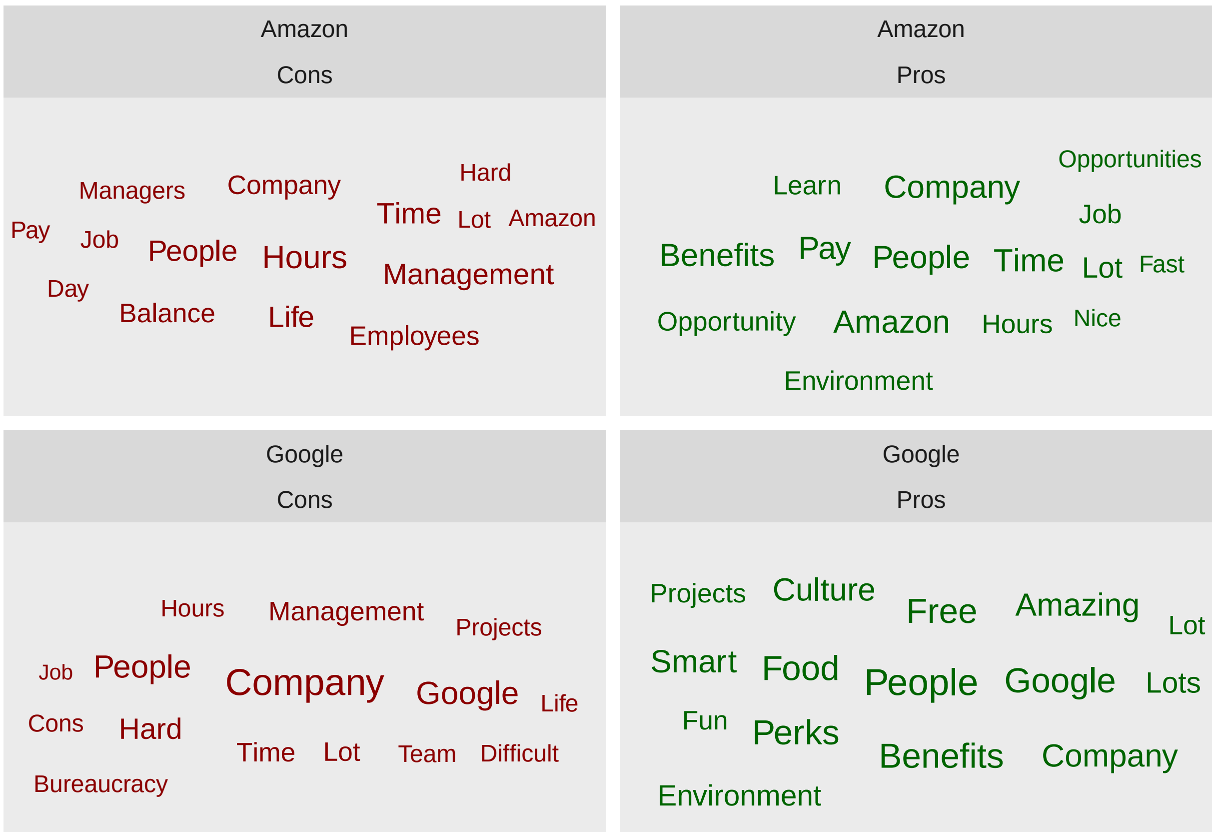
Create a plot to visualize the top word counts with a word cloud. Call **ggplot()** and set the *data* to **top_word_count**, the *label* to **word**, the *size* to **tf_idf**, the *color* to **type**, and the *shape* to **square**. Add a **geom_text_wordlcoud()** layer and set the *show legend* option to **FALSE**. Scale the area with **scale_size_area()** and set the *max size* to **5**. Create facets of **org** and **type** with **facet_wrap()**. Fill the *cons* words *dark red* and *pros* words *dark green*.

Questions 5.3: Which word in *Google cons* and *Google pros* occurs relatively less prominently with respect to *term frequency - inverse document frequency*?

Responses 5.3: *Projects*.

```r
## word cloud
ggplot(
  top_word_count,
  aes(
    label = word,
    size = tf_idf,
    color = type
  ),
  shape = "square"
) +
  geom_text_wordcloud(show.legend = FALSE) +
  scale_size_area(max_size = 5) +
  facet_wrap(
    vars(org, type),
```

```
    scales = "free"
  ) +
  scale_color_manual(
    values = c(
      "darkred",
      "darkgreen"
    )
  )
```



Task 6: Sentiment Analysis

For this task, you will perform sentiment analysis.

Task 6.1

Create a data table named **bing_sent**. Pipe **unigrams** into **inner_join()** to join with the *bing sentiments* by **word**. Change to *title case* **org**, **type**, and **sentiment**. Group by **org** and **type**. Count by the number of **sentiment** values by the formed groups. Calculate the *proportion* (named **prop**) of each **sentiment** value by the formed groups. Remove the groups.

Create plot object named **bing_sent_plot**. Call **ggplot()** and set the *data* to **bing_sent**, the *x-axis* to **sentiment**, the *y-axis* to **prop**, and the *fill* to **type**. Add a **geom_col()** layer and set the *show legend* option to **FALSE**. Add a **geom_text()** layer and map to *label* rounded values of **prop** to *three* digits, position the values in the middle of the bars, and color the values *white*. Create facets of **org** in the rows and **type** in the columns with **facet_grid()**. Fill the *cons* bars *dark red* and *pros* bars *dark green*. Label the axes appropriately. Display the plot.

Questions 6.1: Answer these questions: (1) Do *Amazon* or *Google* employee reviews have a higher proportion of *negative sentiment* for the *cons* comments? (2) Do *Amazon* or *Google* employee reviews have a higher proportion of *positive sentiment* for the *pros* comments?

Responses 6.1: *(1) Amazon (2) Google.*

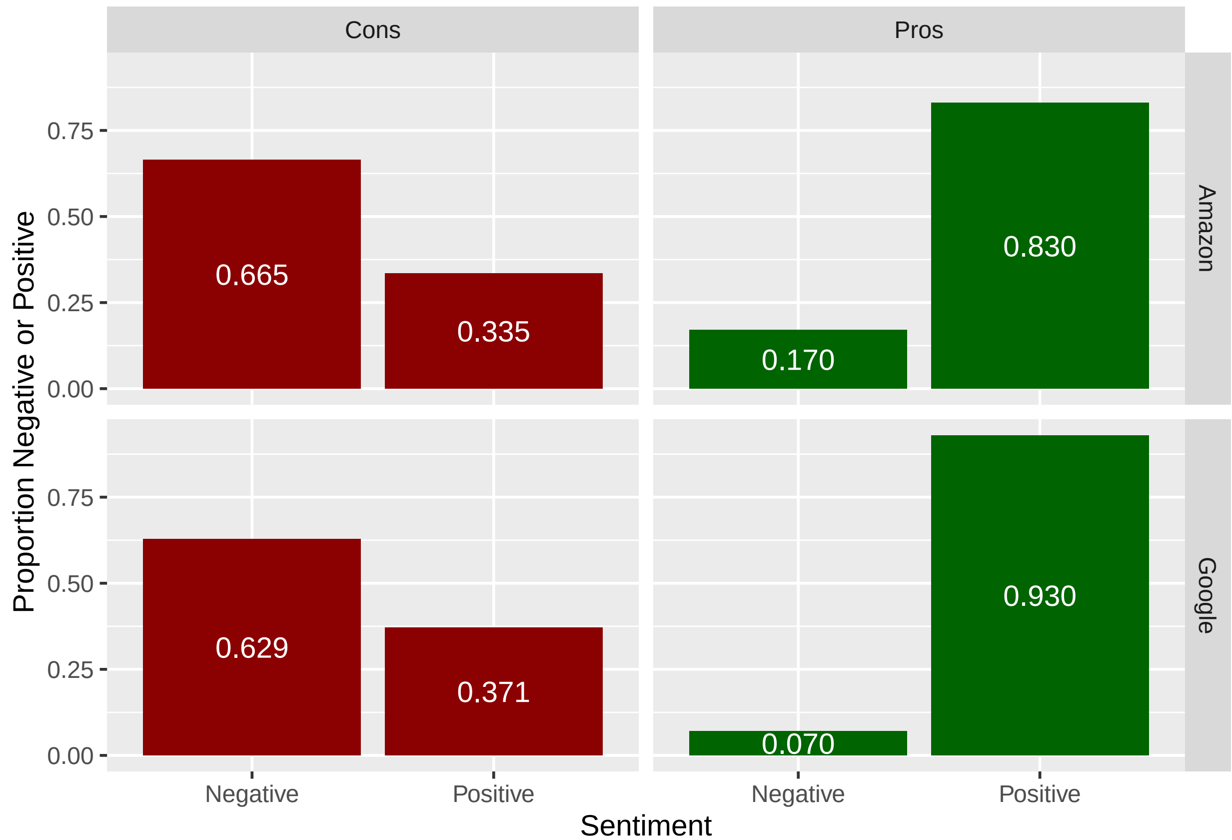
```
### examine dictionary
## bing
get_sentiments("bing") %>%
  ## sample
  slice_sample(n = 20)
```

```
## # A tibble: 20 x 2
##    word           sentiment
##    <chr>          <chr>
##  1 fatal          negative
##  2 bulkiness      negative
##  3 scourge        negative
##  4 persecution    negative
##  5 wows           positive
##  6 discouragement negative
##  7 raptureously   positive
##  8 malcontented   negative
##  9 issues         negative
## 10 famine         negative
## 11 work           positive
## 12 lame-duck      negative
## 13 devilishly     negative
## 14 atrocities     negative
## 15 cautionary     negative
## 16 rantingly      negative
## 17 righteousness  positive
## 18 diligently     positive
## 19 imprecisely    negative
## 20 ploy           negative
```

```
### score sentiments
## call data
bing_sent <- unigrams %>%
  ## inner join
  inner_join(
    # sentiment
    get_sentiments("bing"),
    # key
    by = "word"
  ) %>%
  ## update variables
  mutate(
    # title case
    across(
      # columns
```

```r
      .cols = c(org, type, sentiment),
      # function
      .fns = str_to_title
    )
  ) %>%
  ## group by organization, comment type
  group_by(org, type) %>%
  ## count sentiments
  count(sentiment, name = "count") %>%
  ## add variable
  mutate(
    # proportions
    prop = count / sum(count)
  ) %>%
  ## remove groups
  ungroup()



### plot bing sentiments
## call plot
bing_sent_plot <- ggplot(
  # data
  bing_sent,
  # mapping
  aes(
    # x-axis
    x = sentiment,
    # y-axis
    y = prop,
    # fill
    fill = type
  )
) +
  ## bars
  geom_col(show.legend = FALSE) +
  ## text
  geom_text(
    # mapping
    aes(
      # rounded labels
      label = format(
        # round
        round(
          # variable
          prop,
          # decimals
          digits = 3
        ),
        # digits
        digits = 3
      )
    ),
```

```
    # position text in middle
    position = position_stack(vjust = 0.5),
    # color
    color = "white"
  ) +
  ## facets
  facet_grid(
    # variables
    org ~ type
  ) +
  ## scale fill
  scale_fill_manual(
    # colors
    values = c(
      # cons
      "darkred",
      # pros
      "darkgreen"
    )
  ) +
  ## axes labels
  labs(x = "Sentiment", y = "Proportion Negative or Positive")

## display plot
bing_sent_plot
```



Task 6.2

Create a data table named **afinn_sent**. Pipe **unigrams** into **inner_join()** to join with the *afinn sentiments* by **word**. Change to *title case* **org**, **type**, and **sentiment**. Group by **org** and **type**. Summarize the **value** column by **sum**, **mean**, **sd**, and **median**. Name the resulting columns by function. Drop the groups.

Create plot object named **afinn_sent_plot**. Call **ggplot()** and set the *data* to **afinn_sent**, the *x-axis* to **type**, the *y-axis* to **mean**, and the *fill* to **type**. Add a **geom_col()** layer and set the *show legend* option to **FALSE**. Add a **geom_text()** layer and map to *label* rounded values of **mean** to *three* digits, position the values in the middle of the bars, color the values **skyblue**, set the *size* to **6**, the *font family* to **serif**, and *font face* to **bold**. Create facets of **org** in the columns with **facet_grid()**. Fill the *cons* bars *dark red* and *pros* bars *dark green*. Label the axes appropriately. Display the plot.

Questions 6.2: Answer these questions: (1) Do *Amazon* or *Google* employee reviews have a *lower sentiment value* for the *cons* comments? (2) Do *Amazon* or *Google* employee reviews have a *higher sentiment value* for the *pros* comments??

Responses 6.2: *(1) Amazon (2) Google.*

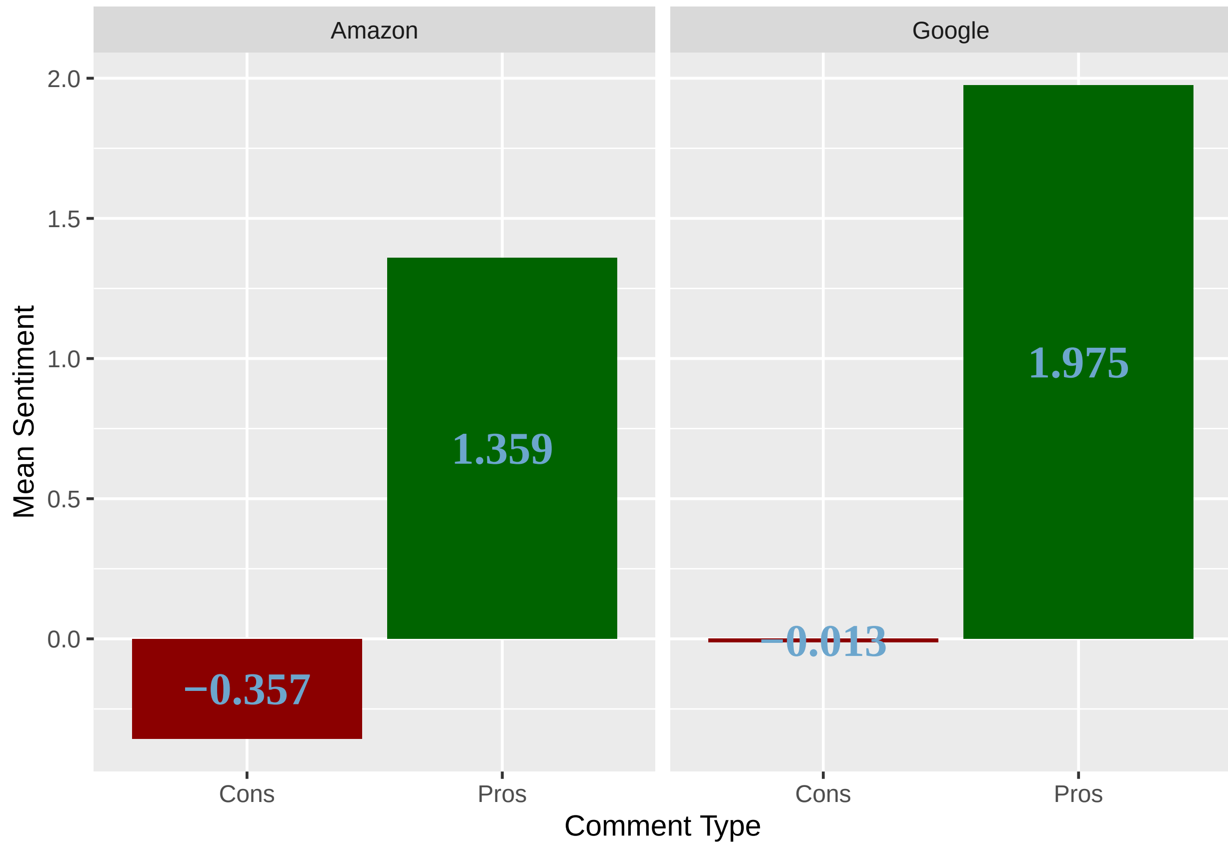
```
### examine dictionary
## afinn
get_sentiments("afinn") %>%
  ## sample
  slice_sample(n = 20)
```

```
## # A tibble: 20 x 2
##    word             value
##    <chr>            <dbl>
##  1 captivated           3
##  2 imperfect           -2
##  3 woebegone           -2
##  4 itchy               -2
##  5 drowned             -2
##  6 immune               1
##  7 certain              1
##  8 fatality            -3
##  9 thwarting           -2
## 10 conflictive         -2
## 11 racists             -3
## 12 heartbroken         -3
## 13 failures            -2
## 14 cocksucker          -5
## 15 victims             -3
## 16 treason             -3
## 17 congratulations      2
## 18 asshole             -4
## 19 prosperous           3
## 20 agonized            -3
```

```
### score sentiments
## call data
afinn_sent <- unigrams %>%
  ## inner join
  inner_join(
    # sentiment
```

```
    get_sentiments("afinn"),
    # key
    by = "word"
  ) %>%
  ## update variables
  mutate(
    # title case
    across(
      # columns
      .cols = c(org, type),
      # function
      .fns = str_to_title
    )
  ) %>%
  ## group by organizations
  group_by(org, type) %>%
  ## summarize value
  summarize(
    # apply functions to variable
    across(
      # columns
      .cols = value,
      # functions
      .fns = list(
        # sum
        sum = sum,
        # mean
        mean = mean,
        # sd
        sd = sd,
        # median
        median = median
      ),
      # name
      .names = "{.fn}"
    ),
    # groups
    .groups = "drop"
  )


### plot afinn sentiments
## call plot
afinn_sent_plot <- ggplot(
  # data
  afinn_sent,
  # mapping
  aes(
    # x-axis
    x = type,
    # y-axis
    y = mean,
    # fill
```

```r
    fill = type
  )
) +
  ## bars
  geom_col(show.legend = FALSE) +
  ## text
  geom_text(
    # mapping
    aes(
      # rounded labels
      label = round(
        # variable
        mean,
        # decimals
        digits = 3
      )
    ),
    # position text in middle
    position = position_stack(vjust = 0.5),
    # color
    color = "skyblue3",
    # size
    size = 6,
    # font family
    family = "serif",
    # font face
    fontface = "bold"
  ) +
  ## facets
  facet_grid(
    # variables
    ~ org
  ) +
  ## scale fill
  scale_fill_manual(
    # colors
    values = c(
      # cons
      "darkred",
      # pros
      "darkgreen"
    )
  ) +
  ## axes labels
  labs(x = "Comment Type", y = "Mean Sentiment")


## display plot
afinn_sent_plot
```



Task 6.3

Create a data table named **nrc_sent**. Pipe **unigrams** into **inner_join()** to join with the *afinn sentiments* by **word**. Change to *title case* **org**, **type**, **word**, and **sentiment**. Count by the number of **org**, **type**, and **sentiment** groups and name the variable **count**. Calculate a new variable named **sentiment_id** reordering **sentiment** within **org** and **type** by **count**.

Create plot object named **nrc_sent_plot**. Call **ggplot()** and set the *data* to **nrc_sent**, the *x-axis* to **sentiment_id**, the *y-axis* to **count**, and the *fill* to **type**. Add a **geom_col()** layer and set the *show legend* option to **FALSE**. Add a **geom_text()** layer and map to *label* rounded values of **count**, position the values in the middle of the bars, color the values **skyblue**, set the *size* to **4**, and *font face* to **bold**. Scale the *x-axis* with **scale_x_reordered()**. Create facets of **org** and **type** with **facet_wrap()**. Fill the *cons* bars *dark red* and *pros* bars *dark green*. Flip the coordinates with **coord_flip()**. Label the axes appropriately. Display the plot.

Questions 6.3: Answer these questions: (1) Do *Amazon* or *Google* employee reviews have *more trust sentiments* in their *pros* comments? (2) Do *Amazon* or *Google* employee reviews have *more anger sentiments* for the *cons* comments??

Responses 6.3: *(1) Amazon (2) Amazon.*

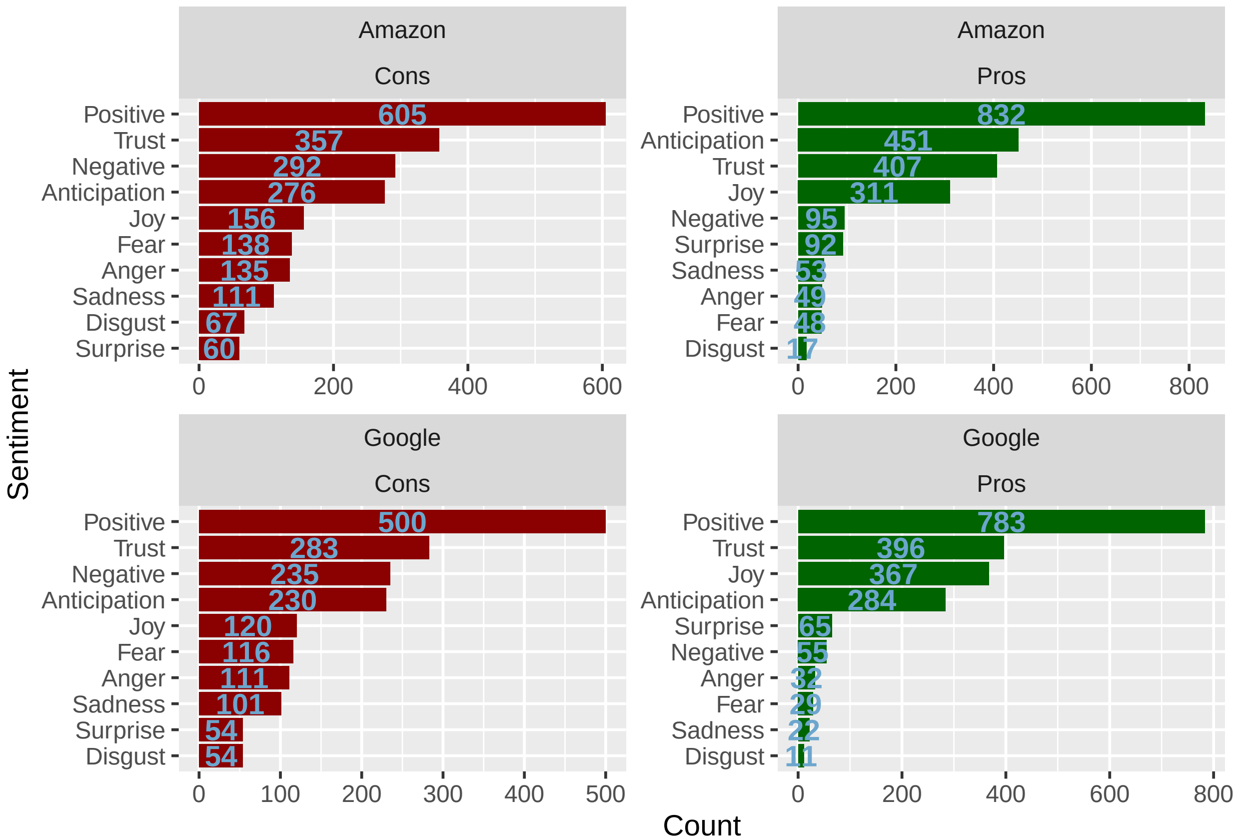
```
## nrc
get_sentiments("nrc") %>%
  slice_sample(n = 20)
```

```
## # A tibble: 20 x 2
##    word           sentiment
```

```
##    <chr>          <chr>
##  1 sanctification trust
##  2 rugged         negative
##  3 interior       disgust
##  4 difficulties   negative
##  5 desolation     negative
##  6 utility        positive
##  7 rot            negative
##  8 misery         anger
##  9 astray         negative
## 10 fain           positive
## 11 congress       trust
## 12 alienation     negative
## 13 trickery       fear
## 14 brute          anger
## 15 frank          positive
## 16 cage           sadness
## 17 attacking      surprise
## 18 insecure       anger
## 19 progress       anticipation
## 20 art            anticipation
```

```r
### score sentiments
nrc_sent <- unigrams %>%
  inner_join(
    get_sentiments("nrc"),
    by = "word"
  ) %>%
  mutate(
    across(
      .cols = c(org, type, word, sentiment),
      .fns = str_to_title
    )
  ) %>%
  count(org, type, sentiment, name = "count") %>%
  mutate(
    sentiment_id = reorder_within(
      sentiment,
      count,
      list(org, type)
    )
  )

### plot nrc sentiments
nrc_sent_plot <- ggplot(
  nrc_sent,
  aes(
    x = sentiment_id,
    y = count,
    fill = type
  )
) +
  geom_col(show.legend = FALSE) +
  geom_text(
```

```
    aes(label = count),
    position = position_stack(vjust = 0.5),
    color = "skyblue3",
    size = 4,
    fontface = "bold"
  ) +
  scale_x_reordered() +
  facet_wrap(
    vars(org, type),
    scales = "free"
  ) +
  scale_fill_manual(
    values = c(
      "darkred",
      "darkgreen"
    )
  ) +
  coord_flip() +
  labs(x = "Sentiment", y = "Count")

## display plot
nrc_sent_plot
```



Task 7: Bigrams

For this task, you will examine bigrams.

Task 7.1

Create a new data table named **bigrams**. Pipe **emp_reviews** into **unnest_tokens()**. Tokenize the **comments** column as *bigrams* using the appropriate inputs, setting the name of the token column to **bigram**. Filter for non-missing values in **bigram**. Count the **bigram** column in **bigrams** while *sorting* the count.

Questions 7.1: What is the most frequent bigram?

Responses 7.1: *To work.*

```
### bigram tokens
bigram <- emp_reviews %>%
  unnest_tokens(
    bigram,
    comment,
    token = "ngrams",
    n = 2
  ) %>%
  filter(
    !is.na(bigram)
  )

## preview
bigram
```

```
## # A tibble: 34,417 x 4
##       id org    type  bigram
##    <int> <fct>  <chr> <chr>
##  1     1 amazon pros  you're surrounded
##  2     1 amazon pros  surrounded by
##  3     1 amazon pros  by smart
##  4     1 amazon pros  smart people
##  5     1 amazon pros  people and
##  6     1 amazon pros  and the
##  7     1 amazon pros  the projects
##  8     1 amazon pros  projects are
##  9     1 amazon pros  are interesting
## 10     1 amazon pros  interesting if
## # ... with 34,407 more rows
```

```
### bigram counts
bigram %>%
  count(bigram, sort = TRUE)
```

```
## # A tibble: 19,792 x 2
##    bigram         n
##    <chr>      <int>
##  1 to work      203
##  2 a lot        162
##  3 lot of       125
##  4 of the       125
##  5 lots of      101
##  6 can be        99
```

```
##  7 the company    97
##  8 in the         95
##  9 if you         94
## 10 work life      87
## # ... with 19,782 more rows
```

Task 7.2

Update **bigrams** by separting the two words in the **bigram** column. Name the new columns **word_1** and **word_2**. Reference the correct separator.

Create a data table named **bigrams_filtered**. Pipe **bigrams** into a *first* filter statement where all of rows with words in **word_1** that are *not* a stop word from **stop_words** are kept. Pipe the result into a *second* filter statement where all of rows with words in **word_2** that are *not* a stop word from **stop_words** are kept.

Create a *table graph* object named **bigrams_tg**. Pipe **bigrams_filtered** into **count()** where the combinations of **word_1** and **word_2** are counted. Name the result **count**. Filter for counts greater then **5**. Convert to a *table graph* using **as_tbl_graph()**.

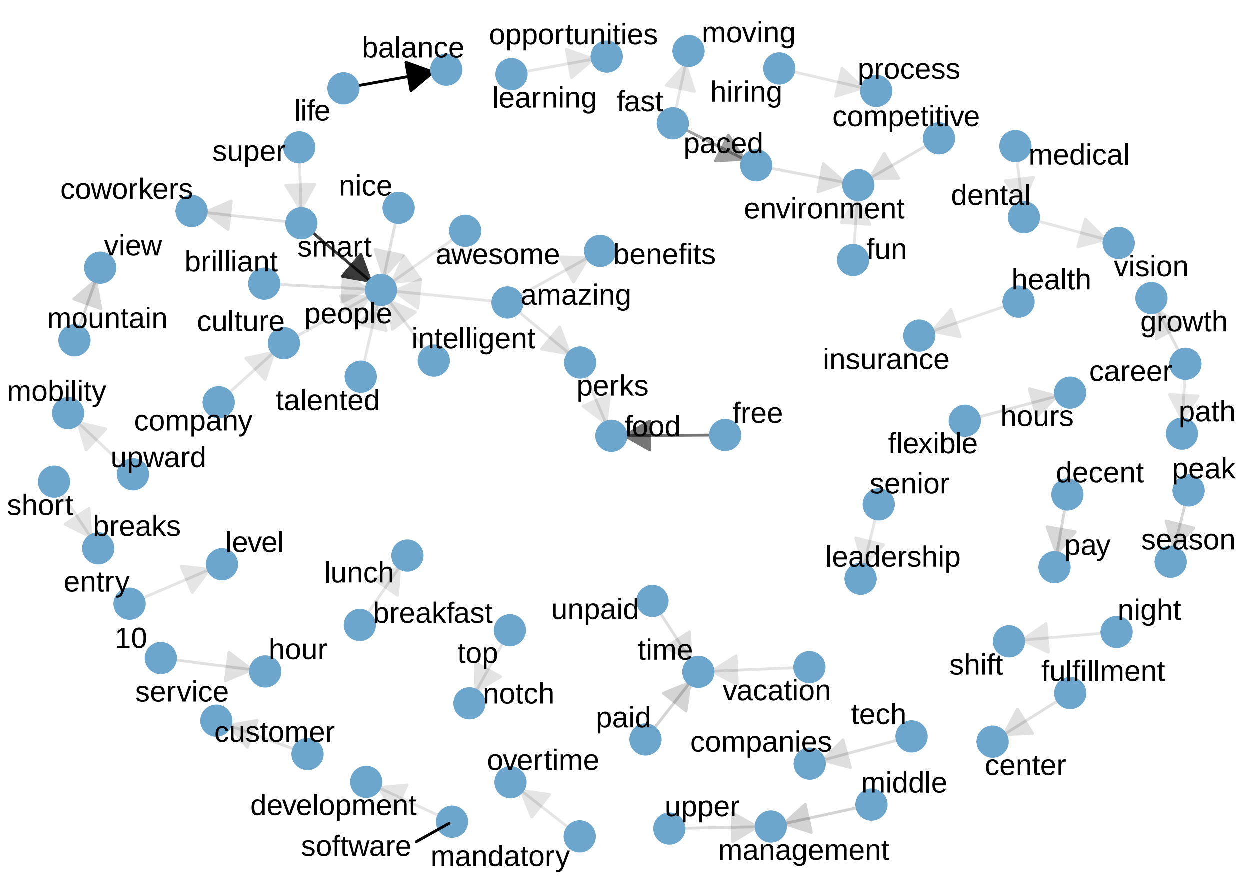
Create a network plot named **bigrams_tg_plot**. Call **ggraph()** and set the *data* to **bigrams_tg** and *layout* to **kk**. Add a **geom_edge_link()** layer, setting **alpha** to **count**, excluding the legend, setting the **arrow** to a *closed triangle*, and setting the **end_cap** to a *circle*. Add a **geom_node_point()** layer, setting the **color** to **skyblue3** and **size** to **5**. Add a **geom_node_text()** layer, setting the **label** to **name** and **repel** to **TRUE**. Remove any theme. Display the plot.

Question 7.2: What *words* describe the *environment* at the two organizations?

Response 7.2: *work life and balance, fast paced, free food* .

```
### separate bigrams
bigram <- bigram %>%
  separate(
    bigram,
    c("word_1", "word_2"),
    sep = " "
  )

### filter for stop words
bigram_filtered <- bigram %>%
  filter(!word_1 %in% stop_words$word) %>%
  filter(!word_2 %in% stop_words$word)

### table graph for bigrams
bigram_tg <- bigram_filtered %>%
  count(word_1, word_2, name = "count", sort = TRUE) %>%
  filter(count >= 5) %>%
  as_tbl_graph()

### plot graph of bigrams
bigram_tg_plot <- ggraph(
  bigram_tg,
  layout = "kk"
) +
  geom_edge_link(
    aes(
```

```
      alpha = count
    ),
    show.legend = FALSE,
    arrow = arrow(
      type = "closed",
      length = unit(0.15, "inches")
    ),
    end_cap = circle(0.07, "inches")
  ) +
  geom_node_point(
    color = "skyblue3",
    size = 5
  ) +
  geom_node_text(
    aes(
      label = name
    ),
    repel = TRUE
  ) +
  theme_void()


## display plot
bigram_tg_plot
```



Task 7.3

Create a new data table named **bigrams_united**. Pipe **bigrams_filtered** into *first* **unite()** where you create a new column named **bigram** from the **word_1** and **word_2** columns. Pipe the result into a *second* **unite()** where you create a new column named **org_type** from the **org** and **type** columns. Change to *title case* the **org_type** and **bigram** columns.

Create a new data tabel named **bigrams_tf_idf**. Pipe **bigrams_united** into **count()**. Count the combinations of **org_type** and **bigram**. Name the new column **count**. Pipe the result into **bind_tf_idf()** and set the term to **bigram**, the document to **org_type**, and counts to **count**. Arrange by *descending* **tf_idf**. Print the result.

Question 7.3: What is the *top bigram* by **tf_idf**?

Response 7.3: *Free food.*

```r
### unite words
bigram_united <- bigram_filtered %>%
  unite(
    bigram,
    word_1, word_2,
    sep = " "
  ) %>%
  unite(
    org_type,
    org, type,
    sep = " "
  ) %>%
  mutate(
    across(
      .cols = c(org_type, bigram),
      .fns = str_to_title
    )
  )

## preview
bigram_united
```

```
## # A tibble: 4,778 x 3
##       id org_type    bigram
##    <int> <chr>       <chr>
##  1     1 Amazon Pros Smart People
##  2     1 Amazon Cons Internal Tools
##  3     1 Amazon Cons Tools Proliferation
##  4     1 Amazon Cons Basic Information
##  5     1 Amazon Cons Learn Understand
##  6     1 Amazon Cons Understand Sql
##  7     1 Amazon Cons Sql Database
##  8     1 Amazon Cons Database Queries
##  9     1 Amazon Cons Actionable Data
## 10     2 Amazon Pros Amazon Hours
## # ... with 4,768 more rows
```

```r
### bigram tf-idf
bigram_tf_idf <- bigram_united %>%
  count(org_type, bigram, name = "count") %>%
  bind_tf_idf(
    bigram,
    org_type,
    count
  ) %>%
  arrange(desc(tf_idf))

## preview
bigram_tf_idf
```

```
## # A tibble: 4,060 x 6
##    org_type    bigram              count      tf   idf  tf_idf
##    <chr>       <chr>               <int>   <dbl> <dbl>   <dbl>
##  1 Google Pros Free Food              42 0.0331  0.693 0.0230
##  2 Google Pros Smart People           42 0.0331  0.288 0.00953
##  3 Amazon Pros Unpaid Time             8 0.00616 1.39  0.00854
##  4 Amazon Cons 10 Hour                 7 0.00567 1.39  0.00786
##  5 Google Pros Amazing Benefits        6 0.00473 1.39  0.00656
##  6 Google Pros Amazing Perks           6 0.00473 1.39  0.00656
##  7 Amazon Pros Vacation Time           6 0.00462 1.39  0.00640
##  8 Google Cons Middle Management       9 0.00921 0.693 0.00639
##  9 Google Cons San Francisco           4 0.00409 1.39  0.00568
## 10 Amazon Cons Mandatory Overtime      5 0.00405 1.39  0.00562
## # ... with 4,050 more rows
```

Task 7.4

Create a new data table named **top_bigrams_count**. Pipe **bigrams_tf_idf** into **group_by()** and form groups via **org_type**. Slice for the *top 10* values of **count** while removing ties. Remove the groups. Calculate a new variable named **bigram_id** reordering **bigram** within **org_type** by **count**.

Create plot object named **top_bigrams_count_plot**. Call **ggplot()** and set the *data* to **top_bigrams_count**, the *x-axis* to **bigram_id**, the *y-axis* to **count**, and the *fill* to **org_type**. Add a **geom_col()** layer and set the *show legend* option to **FALSE**. Add a **geom_text()** layer and map to *label* rounded values of **count**, position the values in the middle of the bars, and color the values **white**. Scale the *x-axis* with **scale_x_reordered()**. Create facets of **org_type** with **facet_wrap()**. Fill the *cons* bars *dark red* and *pros* bars *dark green*. Flip the coordinates with **coord_flip()**. Label the axes appropriately. Display the plot.

Question 7.4: Answer these questions: (1) What is the *most frequent bigram* for *Amazon pros* comments? (2) Are there *more life balance bigrams* for *Google cons* or *Google pros*?

Response 7.4: *(1) Smart People (2) Google .*

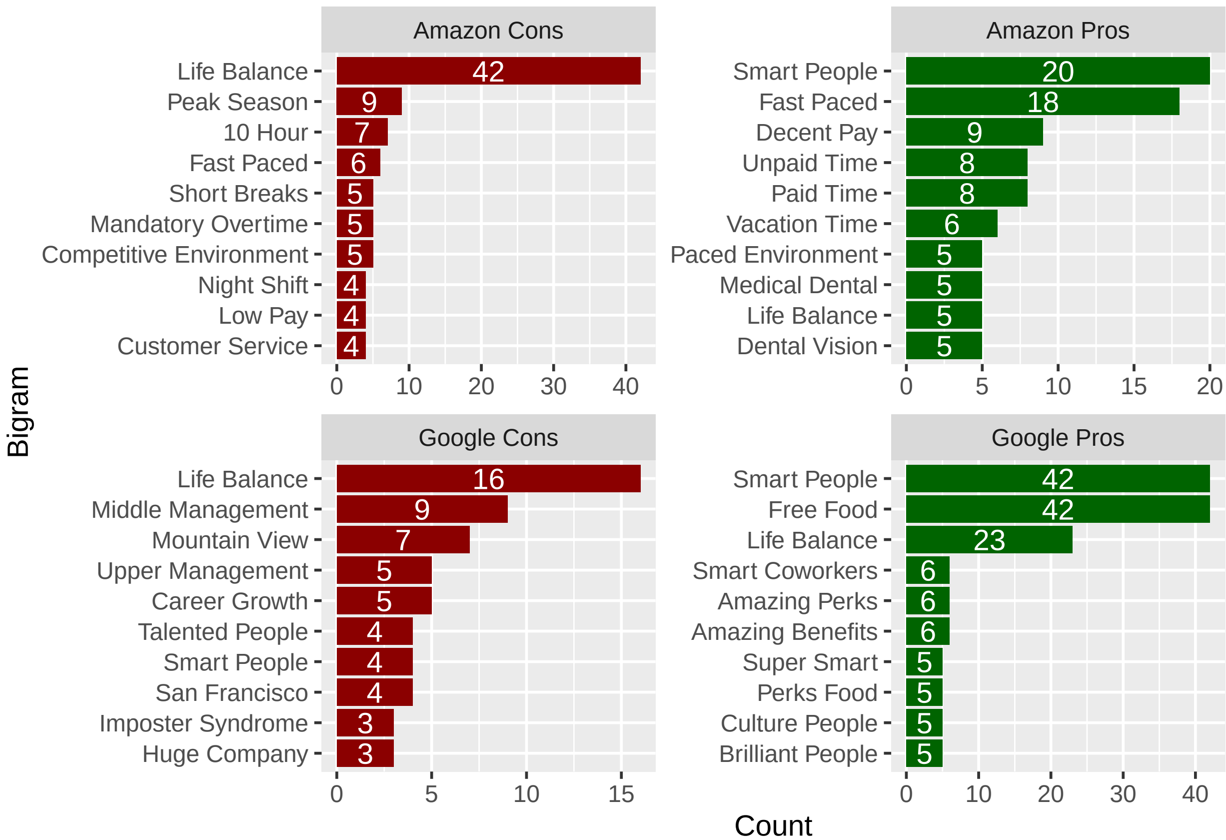
```r
### compare top words by organization, comment type
top_bigram_count <- bigram_tf_idf %>%
  group_by(org_type) %>%
  slice_max(
    count,
    n = 10,
    with_ties = FALSE
```

```r
  ) %>%
  ungroup() %>%
  mutate(
    bigram_id = reorder_within(
      bigram,
      count,
      org_type
    )
  )

## print
top_bigram_count %>%
  print(n = Inf)
```

```
## # A tibble: 40 x 7
##    org_type   bigram          count      tf   idf  tf_idf bigram_id
##    <chr>      <chr>           <int>   <dbl> <dbl>   <dbl> <fct>
##  1 Amazon Co~ Life Balance       42 0.0340  0     0       Life Balance___Amazon ~
##  2 Amazon Co~ Peak Season         9 0.00729 0.693 0.00506 Peak Season___Amazon C~
##  3 Amazon Co~ 10 Hour             7 0.00567 1.39  0.00786 10 Hour___Amazon Cons
##  4 Amazon Co~ Fast Paced          6 0.00486 0     0       Fast Paced___Amazon Co~
##  5 Amazon Co~ Mandatory Ove~      5 0.00405 1.39  0.00562 Mandatory Overtime___A~
##  6 Amazon Co~ Short Breaks        5 0.00405 1.39  0.00562 Short Breaks___Amazon ~
##  7 Amazon Co~ Competitive E~      5 0.00405 0.288 0.00117 Competitive Environmen~
##  8 Amazon Co~ Low Pay             4 0.00324 1.39  0.00449 Low Pay___Amazon Cons
##  9 Amazon Co~ Customer Serv~      4 0.00324 0.693 0.00225 Customer Service___Ama~
## 10 Amazon Co~ Night Shift         4 0.00324 0.693 0.00225 Night Shift___Amazon C~
## 11 Amazon Pr~ Smart People       20 0.0154  0.288 0.00443 Smart People___Amazon ~
## 12 Amazon Pr~ Fast Paced         18 0.0139  0     0       Fast Paced___Amazon Pr~
## 13 Amazon Pr~ Decent Pay          9 0.00693 0.693 0.00480 Decent Pay___Amazon Pr~
## 14 Amazon Pr~ Unpaid Time         8 0.00616 1.39  0.00854 Unpaid Time___Amazon P~
## 15 Amazon Pr~ Paid Time           8 0.00616 0.693 0.00427 Paid Time___Amazon Pros
## 16 Amazon Pr~ Vacation Time       6 0.00462 1.39  0.00640 Vacation Time___Amazon~
## 17 Amazon Pr~ Dental Vision       5 0.00385 1.39  0.00534 Dental Vision___Amazon~
## 18 Amazon Pr~ Medical Dental      5 0.00385 1.39  0.00534 Medical Dental___Amazo~
## 19 Amazon Pr~ Paced Environ~      5 0.00385 0.693 0.00267 Paced Environment___Am~
## 20 Amazon Pr~ Life Balance        5 0.00385 0     0       Life Balance___Amazon ~
## 21 Google Co~ Life Balance       16 0.0164  0     0       Life Balance___Google ~
## 22 Google Co~ Middle Manage~      9 0.00921 0.693 0.00639 Middle Management___Go~
## 23 Google Co~ Mountain View       7 0.00716 0.693 0.00497 Mountain View___Google~
## 24 Google Co~ Career Growth       5 0.00512 0.693 0.00355 Career Growth___Google~
## 25 Google Co~ Upper Managem~      5 0.00512 0.288 0.00147 Upper Management___Goo~
## 26 Google Co~ San Francisco       4 0.00409 1.39  0.00568 San Francisco___Google~
## 27 Google Co~ Talented Peop~      4 0.00409 0.693 0.00284 Talented People___Goog~
## 28 Google Co~ Smart People        4 0.00409 0.288 0.00118 Smart People___Google ~
## 29 Google Co~ Huge Company        3 0.00307 1.39  0.00426 Huge Company___Google ~
## 30 Google Co~ Imposter Synd~      3 0.00307 1.39  0.00426 Imposter Syndrome___Go~
## 31 Google Pr~ Free Food          42 0.0331  0.693 0.0230  Free Food___Google Pros
## 32 Google Pr~ Smart People       42 0.0331  0.288 0.00953 Smart People___Google ~
## 33 Google Pr~ Life Balance       23 0.0181  0     0       Life Balance___Google ~
## 34 Google Pr~ Amazing Benef~      6 0.00473 1.39  0.00656 Amazing Benefits___Goo~
## 35 Google Pr~ Amazing Perks       6 0.00473 1.39  0.00656 Amazing Perks___Google~
## 36 Google Pr~ Smart Coworke~      6 0.00473 0.693 0.00328 Smart Coworkers___Goog~
```

```
## 37 Google Pr~ Perks Food          5 0.00394 1.39  0.00547 Perks Food___Google Pr~
## 38 Google Pr~ Culture People      5 0.00394 0.693 0.00273 Culture People___Googl~
## 39 Google Pr~ Super Smart         5 0.00394 0.693 0.00273 Super Smart___Google P~
## 40 Google Pr~ Brilliant Peo~      5 0.00394 0.288 0.00113 Brilliant People___Goo~
```

```r
### visualize top word counts
top_bigram_count_plot <- ggplot(
  top_bigram_count,
  aes(
    x = bigram_id,
    y = count,
    fill = org_type
  )
) +
  geom_col(show.legend = FALSE) +
  geom_text(
    aes(
      label = round(
        count
      )
    ),
    position = position_stack(vjust = 0.5),
    color = "white"
  ) +
  scale_x_reordered() +
  facet_wrap(
    vars(org_type),
    scales = "free"
  ) +
  scale_fill_manual(
    values = c(
      "darkred", "darkgreen",
      "darkred", "darkgreen"
    )
  ) +
  coord_flip() +
  labs(x = "Bigram", y = "Count")

## display plot
top_bigram_count_plot
```



Task 8: Counting and Correlating Pairs of Words

For this task, you will count and correlate pairs of words.

Task 8.1

Create a new data table named **word_pairs**. Pipe **unigrams** into **unite()**. Create a new column named **id_org_type** from **id**, **org**, and **type**. Change to *title case* **id_org_type** and **word**. Pipe the result to **pairwise_count()** to count **word** by **id_org_type** and sorting the result.

Create a new data table named **word_pairs_tg**. Pipe **word_pairs** into **rename()** to rename **n** to **count**. Filter by counts greater than or equal to **15**. Convert to a *table graph* with **as_tbl_graph()**.

Create a network plot named **word_pairs_tg_plot**. Call **ggraph()** and set the *data* to **word_pairs_tg** and *layout* to **kk**. Add a **geom_edge_link()** layer, setting **alpha** to **count**, and excluding the legend. Add a **geom_node_point()** layer, setting the **color** to **skyblue3** and **size** to **5**. Add a **geom_node_text()** layer, setting the **label** to **name** and **repel** to **TRUE**. Remove any theme. Display the plot.

Questions 8.1: Answer these questions: (1) Which *word* pairs with the most other words in the plot? (2) With what *word* does *decent* pair in the plot?

Responses 8.1: *(1) People (2) Pay.*

```
### count word pairs
word_pairs <- unigrams %>%
  unite(
    id_org_type,
```

```
    id, org, type,
    sep = " "
  ) %>%
  mutate(
    across(
      .cols = c(id_org_type, word),
      .fns = str_to_title
    )
  ) %>%
  pairwise_count(
    word,
    id_org_type,
    sort = TRUE
  )
```

```
## Warning: 'distinct_()' was deprecated in dplyr 0.7.0.
## Please use 'distinct()' instead.
## See vignette('programming') for more help
```

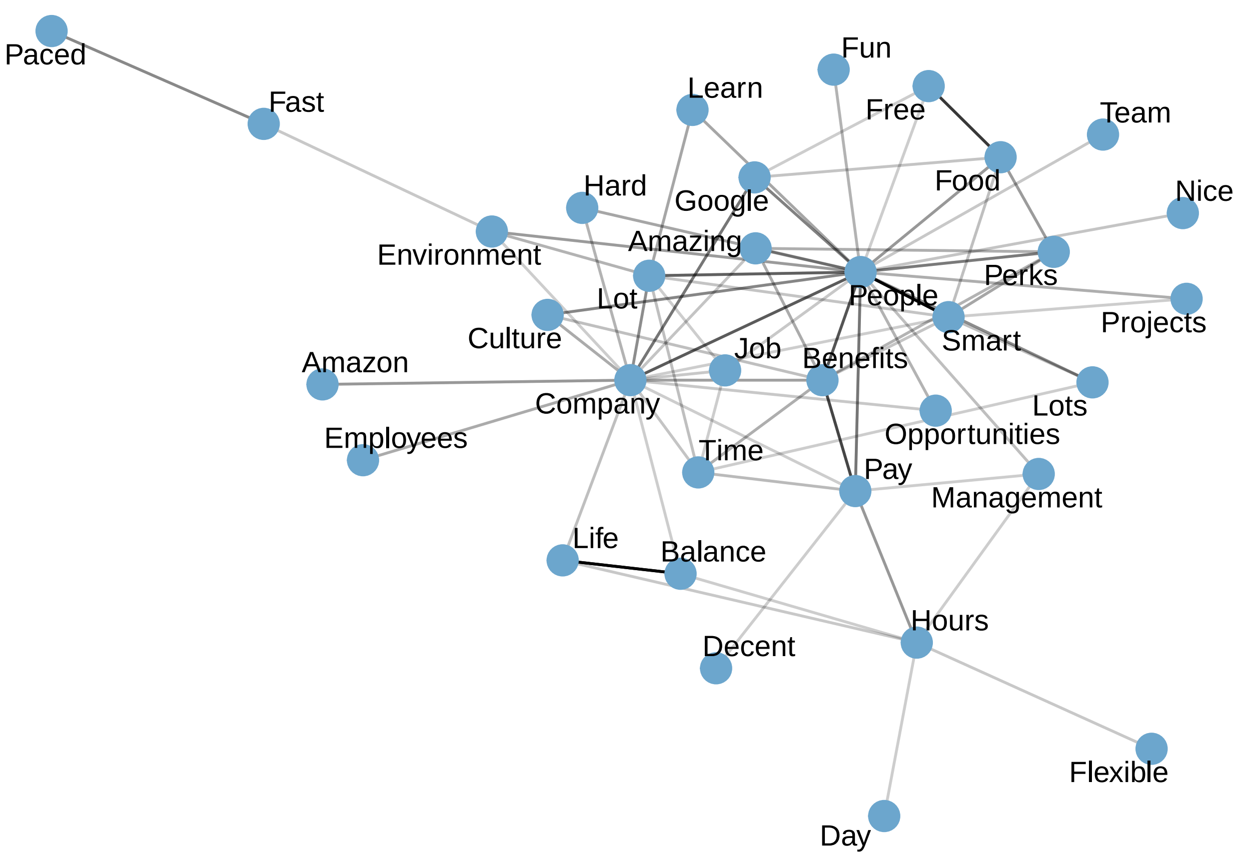
```
## Warning: 'tbl_df()' was deprecated in dplyr 1.0.0.
## Please use 'tibble::as_tibble()' instead.
```

```
## preview
word_pairs
```

```
## # A tibble: 125,188 x 3
##    item1    item2        n
##    <chr>    <chr>    <dbl>
##  1 Balance  Life        90
##  2 Life     Balance     90
##  3 People   Smart       81
##  4 Smart    People      81
##  5 Free     Food        51
##  6 Food     Free        51
##  7 Benefits Pay         47
##  8 Pay      Benefits    47
##  9 Benefits People      42
## 10 People   Benefits    42
## # ... with 125,178 more rows
```

```
### table graph for word pairs
word_pairs_tg <- word_pairs %>%
  rename(count = n) %>%
  filter(count >= 15) %>%
  as_tbl_graph()

### plot graph of bigrams
word_pairs_tg_plot <- ggraph(
  word_pairs_tg,
  layout = "kk"
) +
  geom_edge_link(
```

```
    aes(
      alpha = count
    ),
    show.legend = FALSE
  ) +
  geom_node_point(
    color = "skyblue3",
    size = 5
  ) +
  geom_node_text(
    aes(
      label = name
    ),
    repel = TRUE
  ) +
  theme_void()

## display plot
word_pairs_tg_plot
```



Task 8.2

Create a new data table named **word_cors**. Pipe **unigrams** into **unite()**. Create a new column named **id_org_type** from **id**, **org**, and **type**. Change to *title case* **id_org_type** and **word**. Group by **word**. Filter for counts greater than or equal to **25**. Pipe the result to **pairwise_cor()** to count **word** by

id_org_type and sorting the result.

Create a new data table named **word_cors_tg**. Pipe **word_cors** into **filter()** to filter by *abosulte correlations* greater than or equal to **0.08**. Create a new column named **cor_type** that indicates whether a correlation is *positive* or *negative*. Convert to a *table graph* with **as_tbl_graph()**.

Create a network plot named **word_cors_tg_plot**. Call **ggraph()** and set the *data* to **word_cors_tg** and *layout* to **kk**. Add a **geom_edge_link()** layer, setting **alpha** to **correlation** and **color** to **cor_type**, excluding the legend, and setting **width** to **2**. Add a **geom_node_point()** layer, setting the **color** to **skyblue3** and **size** to **5**. Add a **geom_node_text()** layer, setting the **label** to **name** and **repel** to **TRUE**. Color the edges *red* and *green* for *negative* and *positive* correlations, respectively. Remove any theme. Display the plot.

Questions 8.2: Are the words *atomsphere* and *workers positively* or *negativel* correlated?

Responses 8.2: *Positively correlated.*

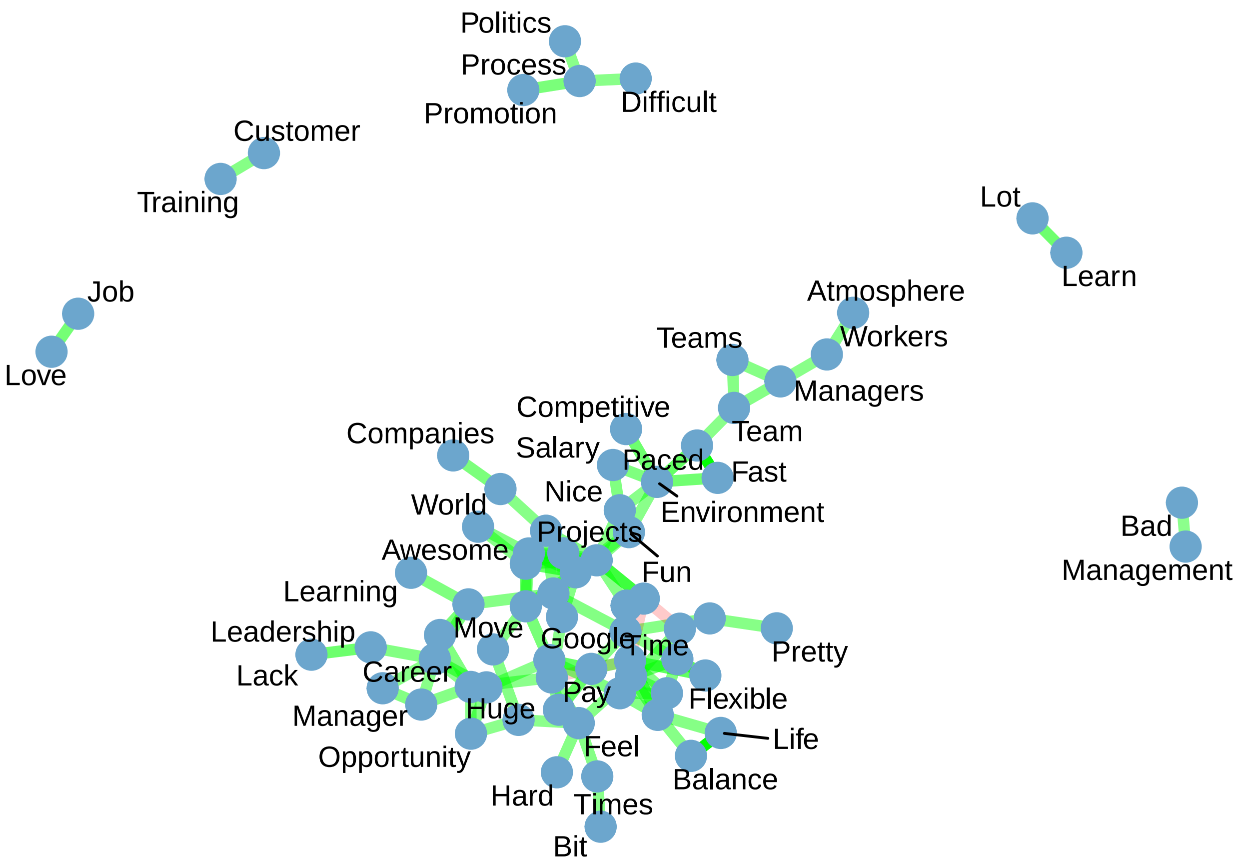
```
### word correlations
word_cors <- unigrams %>%
  unite(
    id_org_type,
    id, org, type,
    sep = " "
  ) %>%
  mutate(
    across(
      .cols = c(id_org_type, word),
      .fns = str_to_title
    )
  ) %>%
  group_by(word) %>%
  filter(
    n() >= 25
  ) %>%
  pairwise_cor(
    word,
    id_org_type,
    sort = TRUE
  )

## preview
word_cors
```

```
## # A tibble: 6,320 x 3
##    item1   item2   correlation
##    <chr>   <chr>         <dbl>
##  1 Balance Life          0.885
##  2 Life    Balance       0.885
##  3 Paced   Fast          0.722
##  4 Fast    Paced         0.722
##  5 Free    Food          0.614
##  6 Food    Free          0.614
##  7 People  Smart         0.347
##  8 Smart   People        0.347
##  9 Week    Days          0.307
## 10 Days    Week          0.307
```

```
## # ... with 6,310 more rows
```

```
### table graph for word pairs
word_cors_tg <- word_cors %>%
  filter(abs(correlation) >= 0.08) %>%
  mutate(
    cor_type = if_else(
      correlation >= 0,
      "positive",
      "negative"
    )
  ) %>%
  as_tbl_graph()



### plot graph of bigrams
word_cors_tg_plot <- ggraph(
  word_cors_tg,
  layout = "kk"
) +
  geom_edge_link(
    aes(
      alpha = correlation,
      color = cor_type
    ),
    show.legend = FALSE,
    width = 2
  ) +
  geom_node_point(
    color = "skyblue3",
    size = 5
  ) +
  geom_node_text(
    aes(
      label = name
    ),
    repel = TRUE,
    check_overlap = TRUE
  ) +
  scale_edge_color_manual(
    values = c("red", "green")
  ) +
  theme_void()



## display plot
word_cors_tg_plot
```

```
## Warning: ggrepel: 25 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



Task 9: Topic Modeling

For this task, you will perform topic modeling via *latent Dirichlet allocation* (LDA).

Task 9.1

Create a document-term matrix named **org_type_dtm**. Pipe **unigrams** into **unite()**. Create a new column named **id_org_type** from **id**, **org**, and **type**. Change to *title case* **id_org_type** and **word**. Count by **id_org_type** and **word** and name the count variable **count**. Remove the groups. Pipe the result into **cast_dtm()** with **id_org_type** as the documents, **word** as the token, and **count** as the frequencies of words. Print **org_type_dtm**.

Use **LDA()** to create a topic model with **8** topics on **org_type_dtm**. Set the random seed to **101**.

Questions 9.1: Answer these questions: (1) How many total *documents* are there in the document-term matrix? (2) How many *non-sparse* entries are there in the document-term matrix?

Responses 9.1: *(1) 1958 (2) 3231.*

```
### create document-term matrix
org_type_dtm <- unigrams %>%
  unite(
    id_org_type,
    id, org, type,
    sep = " "
  ) %>%
```

```r
  mutate(
    across(
      .cols = c(id_org_type, word),
      .fns = str_to_title
    ),
    id = NULL
  ) %>%
  count(id_org_type, word, name = "count", sort = TRUE) %>%
  ungroup() %>%
  cast_dtm(
    id_org_type,
    word,
    count
  )

## first preview
org_type_dtm
```

```
## <<DocumentTermMatrix (documents: 1958, terms: 3231)>>
## Non-/sparse entries: 13925/6312373
## Sparsity           : 100%
## Maximal term length: 19
## Weighting          : term frequency (tf)
```

```r
## second preview
# call data
org_type_dtm %>%
  # convert to matrix
  as.matrix() %>%
  # preview subset of terms
  .[31:50, 11:20]
```

```
##                  Terms
## Docs               Busy Aws Job Shift Company Employees Amazon Lot Day Growing
##   365 Amazon Pros     0   0   0     0       1         0      1   0   0       0
##   371 Amazon Pros     0   0   0     0       1         1      1   0   0       0
##   380 Amazon Cons     0   0   0     0       2         1      2   0   0       0
##   380 Amazon Pros     0   0   1     0       0         0      3   0   0       0
##   40 Amazon Cons      0   0   0     0       0         0      0   0   0       0
##   401 Amazon Pros     0   0   0     0       0         0      0   0   0       0
##   426 Amazon Pros     0   1   0     0       3         0      0   0   0       0
##   434 Amazon Cons     0   0   1     0       0         1      0   1   0       0
##   436 Amazon Cons     0   0   0     0       0         0      0   0   3       0
##   476 Amazon Cons     0   0   0     0       0         0      0   0   0       0
##   480 Amazon Cons     0   0   0     1       0         0      1   1   0       0
##   488 Amazon Cons     0   0   0     0       0         0      0   0   0       0
##   501 Google Pros     0   0   0     0       1         0      0   0   0       0
##   502 Google Pros     0   0   0     0       0         0      0   0   1       0
##   544 Google Cons     0   0   0     0       0         0      0   0   0       0
##   547 Google Cons     0   0   0     0       1         0      0   0   0       0
##   58 Amazon Pros      0   2   0     0       1         0      1   0   0       0
##   610 Google Pros     0   0   0     0       3         0      0   0   0       0
##   631 Google Pros     0   0   0     0       0         0      0   0   1       0
```

```
##    634 Google Cons     0    0    0      0        0           0       0    0    0         0
```

```
### fit LDA
## save as object
org_type_lda <- LDA(
  # dtm
  org_type_dtm,
  # number of topics
  k = 8,
  # controls
  control = list(seed = 101)
)
```

Task 9.2

Create a data table named **lda_word_prob**. Apply **tidy()** to **org_type_lda** and set the **matrix** to **beta**.

Create a new data table named **lda_top_terms**. Pipe **lda_word_prob** into **group_by()** and form groups via **topic**. Slice for the *top 5* values of **beta** while removing ties. Remove the groups. Calculate a new variable named **term_id** reordering **term** within **topic** by **beta**. Convert **topic** to a *factor*.

Create plot object named **lda_top_terms_plot**. Call **ggplot()** and set the *data* to **lda_top_terms**, the *x-axis* to **term_id**, the *y-axis* to **beta**, and the *fill* to **topic**. Add a **geom_col()** layer and set the *show legend* option to **FALSE**. Add a **geom_text()** layer and map to *label* rounded values of **beta** to **3** digits, position the values in the middle of the bars, color the values **black**, and use the **bold** font face. Scale the *x-axis* with **scale_x_reordered()**. Create facets of **topic** with **facet_wrap()**. Flip the coordinates with **coord_flip()**. Label the axes appropriately. Display the plot.

Create a new data table named **word_class**. Apply **augment()** to **org_type_lda** with **data** set to **org_type_dtm**. Rename **.topic** to **topic**. Convert **topic** to a *factor*.

Question 9.2: Answer these questions: (1) What is the *most probable word* for the *fourth topic*? (2) Does the word *people* more likely to belong to the *third* or *sixth* topic?

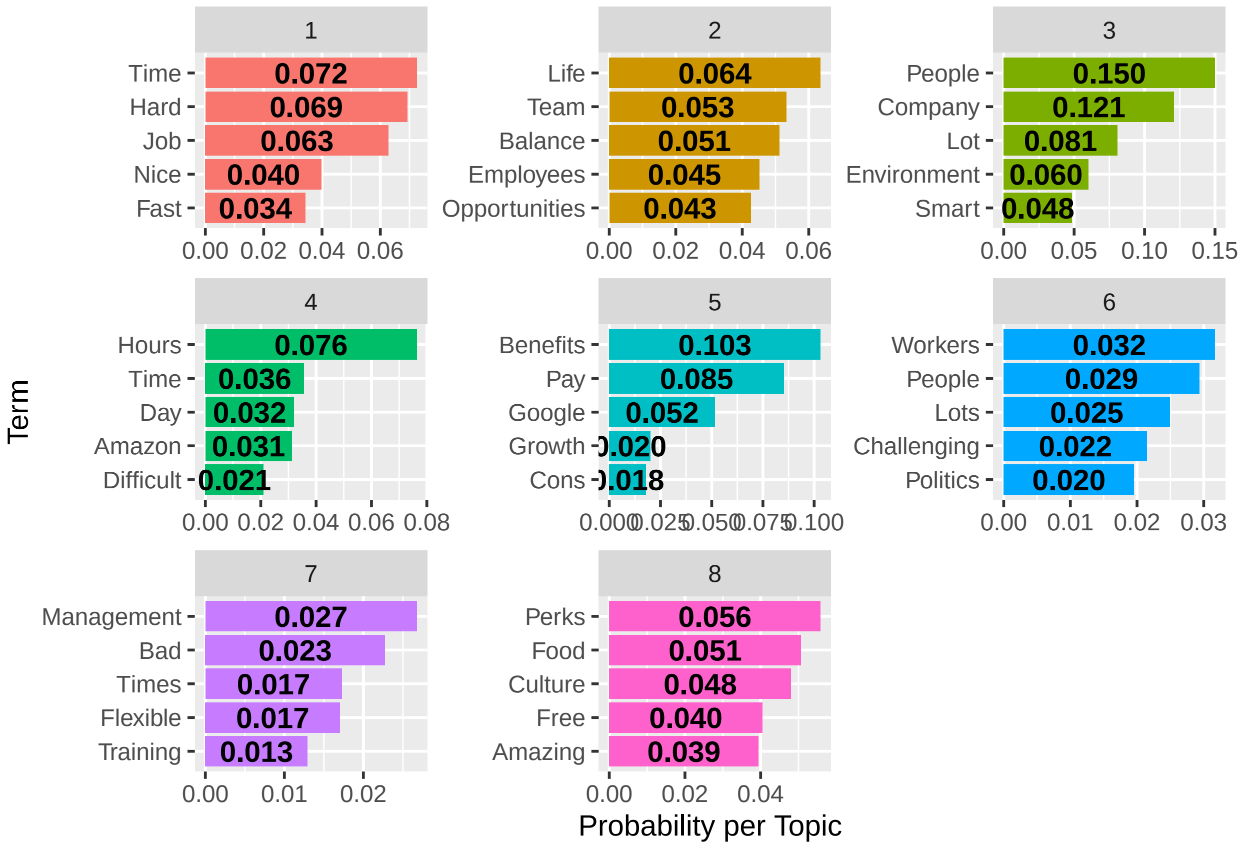
Response 9.2: *(1) Hours (2) the third topic.*

```
### extract word probabilities per topic
lda_word_prob <- tidy(
  org_type_lda,
  matrix = "beta"
)

## finding most probable terms
lda_top_terms <- lda_word_prob %>%
  group_by(topic) %>%
  slice_max(
    beta,
    n = 5,
    with_ties = FALSE
  ) %>%
  ungroup() %>%
  mutate(
    term_id = reorder_within(
      term,
```

```r
      beta,
      topic
    ),
    topic = as_factor(topic)
  )

## print
lda_top_terms %>%
  print(n = Inf)
```

```
## # A tibble: 40 x 4
##    topic term           beta term_id
##    <fct> <chr>         <dbl> <fct>
##  1 1     Time          0.0725 Time___1
##  2 1     Hard          0.0692 Hard___1
##  3 1     Job           0.0628 Job___1
##  4 1     Nice          0.0399 Nice___1
##  5 1     Fast          0.0343 Fast___1
##  6 2     Life          0.0636 Life___2
##  7 2     Team          0.0532 Team___2
##  8 2     Balance       0.0512 Balance___2
##  9 2     Employees     0.0452 Employees___2
## 10 2     Opportunities 0.0425 Opportunities___2
## 11 3     People        0.150  People___3
## 12 3     Company       0.121  Company___3
## 13 3     Lot           0.0807 Lot___3
## 14 3     Environment   0.0601 Environment___3
## 15 3     Smart         0.0483 Smart___3
## 16 4     Hours         0.0763 Hours___4
## 17 4     Time          0.0356 Time___4
## 18 4     Day           0.0320 Day___4
## 19 4     Amazon        0.0312 Amazon___4
## 20 4     Difficult     0.0210 Difficult___4
## 21 5     Benefits      0.103  Benefits___5
## 22 5     Pay           0.0852 Pay___5
## 23 5     Google        0.0516 Google___5
## 24 5     Growth        0.0201 Growth___5
## 25 5     Cons          0.0179 Cons___5
## 26 6     Workers       0.0317 Workers___6
## 27 6     People        0.0293 People___6
## 28 6     Lots          0.0249 Lots___6
## 29 6     Challenging   0.0215 Challenging___6
## 30 6     Politics      0.0196 Politics___6
## 31 7     Management    0.0267 Management___7
## 32 7     Bad           0.0227 Bad___7
## 33 7     Times         0.0173 Times___7
## 34 7     Flexible      0.0170 Flexible___7
## 35 7     Training      0.0129 Training___7
## 36 8     Perks         0.0558 Perks___8
## 37 8     Food          0.0507 Food___8
## 38 8     Culture       0.0481 Culture___8
## 39 8     Free          0.0405 Free___8
## 40 8     Amazing       0.0394 Amazing___8
```

```r
### visualize top term probabilities
lda_top_terms_plot <- ggplot(
  lda_top_terms,
  aes(
    x = term_id,
    y = beta,
    fill = topic
  )
) +
  geom_col(show.legend = FALSE) +
  geom_text(
    aes(
      label = format(
        round(
          beta,
          digits = 3
        ),
        digits = 3
      )
    ),
    position = position_stack(vjust = 0.5),
    color = "black",
    fontface = "bold"
  ) +
  scale_x_reordered() +
  facet_wrap(
    vars(topic),
    scales = "free"
  ) +
  coord_flip() +
  labs(x = "Term", y = "Probability per Topic")

## display plot
lda_top_terms_plot
```




```r
### assign words to topics
word_class <- augment(
  org_type_lda,
  data = org_type_dtm
) %>%
  rename(topic = .topic) %>%
  mutate(
    topic = as_factor(topic)
  )

## preview
word_class
```

```
## # A tibble: 13,925 x 4
##    document         term  count topic
##    <chr>            <chr> <dbl> <fct>
##  1 909 Google Pros Money     7 8
##  2 290 Amazon Pros Money     1 8
##  3 48 Amazon Cons  Money     1 8
##  4 688 Google Cons Money     1 8
##  5 91 Amazon Pros  Money     1 8
##  6 190 Amazon Pros Money     1 8
##  7 20 Amazon Pros  Money     1 8
##  8 242 Amazon Cons Money     1 8
##  9 287 Amazon Cons Money     1 8
## 10 292 Amazon Pros Money     1 8
## # ... with 13,915 more rows
```

Task 9.3

Create a data table named **lda_topic_prob**. Apply **tidy()** to **org_type_lda** and set the **matrix** to **gamma**. Convert **topic** to a *factor* and reverse its levels with **fct_rev()**. Separete the column **document** into columns **id**, **org**, and **type**.

Create a new data table named **lda_summ_topics**. Pipe **lda_topic_prob** into **group_by()** and form groups via **id**, **org**, and **type**. Slice for the *top* value of **gamma**. Group by **org** and **type**. Count by **topic** and name the count **topic_count**. Remove the groups. Calculate a new variable named **topic_id** reordering **topic** within **org** and **type** by **topic_count**.

Create plot object named **lda_summ_topics_plot**. Call **ggplot()** and set the *data* to **lda_summ_topics**, the *x-axis* to **topic_id**, the *y-axis* to **topic_count**, and the *fill* to **type**. Add a **geom_col()** layer and set the *show legend* option to **FALSE**. Add a **geom_text()** layer and map to *label* rounded values of **beta** to **2** digits, position the values in the middle of the bars, color the values **skyblue3**, and use the **bold** font face. Scale the *x-axis* with **scale_x_reordered()**. Color the bars by *cons* and *pros* in *dark red* and *dark green*, respectively. Create facets of **org** and **type** with **facet_wrap()**. Flip the coordinates with **coord_flip()**. Label the axes appropriately. Display the plot.

Create a new data table named **topic_class**. Pipe **lda_topic_prob** into **group_by()** and form groups via **id**, **org**, and **type**. Slice by **gamma**. Remove groups.

Question 9.3: Answer these questions: (1) What is the *most frequent topic* for *Google cons*? (2) What is the *most frequent topic* for *Amazon pros*?

Response 9.3: *(1)topic 5 (2) topic 3.*

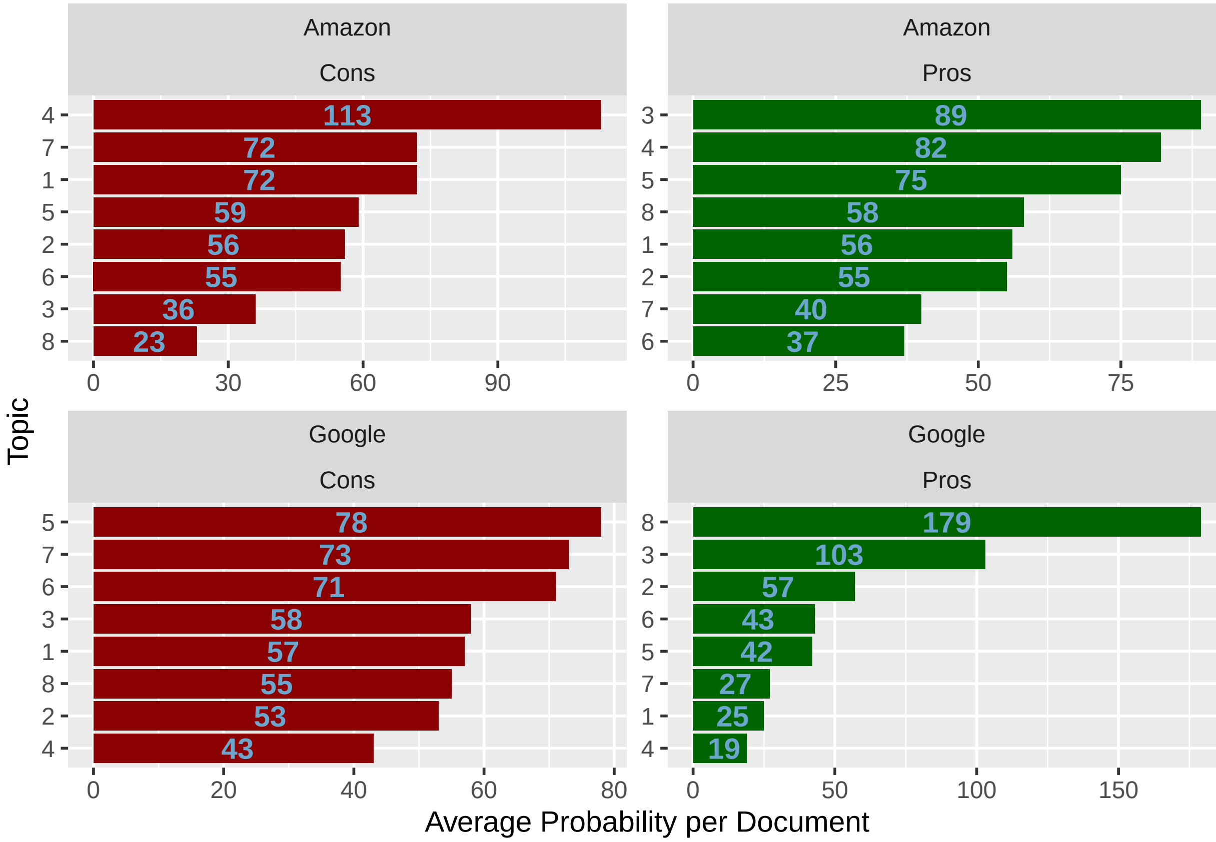
```
### extract topic probabilities per document
lda_topic_prob <- tidy(
  org_type_lda,
  matrix = "gamma"
) %>%
  mutate(
    topic = as_factor(topic),
    topic = fct_rev(topic)
  ) %>%
  separate(
    document,
    c("id", "org", "type"),
    sep = " "
  )

### summarize topic probabilities
lda_summ_topics <- lda_topic_prob %>%
  group_by(id, org, type) %>%
  slice_max(
    gamma,
    n = 1
  ) %>%
  group_by(org, type) %>%
  count(topic, name = "topic_count") %>%
  ungroup() %>%
  mutate(
    topic_id = reorder_within(
      topic,
      topic_count,
      list(org, type)
```

```r
    )
  )

## preview
lda_summ_topics %>%
  print(n = Inf)
```

```
## # A tibble: 32 x 5
##    org    type  topic topic_count topic_id
##    <chr>  <chr> <fct>       <int> <fct>
##  1 Amazon Cons  8              23 8___Amazon___Cons
##  2 Amazon Cons  7              72 7___Amazon___Cons
##  3 Amazon Cons  6              55 6___Amazon___Cons
##  4 Amazon Cons  5              59 5___Amazon___Cons
##  5 Amazon Cons  4             113 4___Amazon___Cons
##  6 Amazon Cons  3              36 3___Amazon___Cons
##  7 Amazon Cons  2              56 2___Amazon___Cons
##  8 Amazon Cons  1              72 1___Amazon___Cons
##  9 Amazon Pros  8              58 8___Amazon___Pros
## 10 Amazon Pros  7              40 7___Amazon___Pros
## 11 Amazon Pros  6              37 6___Amazon___Pros
## 12 Amazon Pros  5              75 5___Amazon___Pros
## 13 Amazon Pros  4              82 4___Amazon___Pros
## 14 Amazon Pros  3              89 3___Amazon___Pros
## 15 Amazon Pros  2              55 2___Amazon___Pros
## 16 Amazon Pros  1              56 1___Amazon___Pros
## 17 Google Cons  8              55 8___Google___Cons
## 18 Google Cons  7              73 7___Google___Cons
## 19 Google Cons  6              71 6___Google___Cons
## 20 Google Cons  5              78 5___Google___Cons
## 21 Google Cons  4              43 4___Google___Cons
## 22 Google Cons  3              58 3___Google___Cons
## 23 Google Cons  2              53 2___Google___Cons
## 24 Google Cons  1              57 1___Google___Cons
## 25 Google Pros  8             179 8___Google___Pros
## 26 Google Pros  7              27 7___Google___Pros
## 27 Google Pros  6              43 6___Google___Pros
## 28 Google Pros  5              42 5___Google___Pros
## 29 Google Pros  4              19 4___Google___Pros
## 30 Google Pros  3             103 3___Google___Pros
## 31 Google Pros  2              57 2___Google___Pros
## 32 Google Pros  1              25 1___Google___Pros
```

```r
### visualize topic probabilities
lda_summ_topics_plot <- ggplot(
  lda_summ_topics,
  aes(
    x = topic_id,
    y = topic_count,
    fill = type
  )
) +
  geom_col(show.legend = FALSE) +
```

```r
  geom_text(
    aes(
      label = format(
        round(
          topic_count,
          digits = 2
        ),
        digits = 2
      )
    ),
    position = position_stack(vjust = 0.5),
    color = "skyblue3",
    fontface = "bold"
  ) +
  scale_x_reordered() +
  scale_fill_manual(
    values = c(
      "darkred", "darkgreen",
      "darkred", "darkgreen"
    )
  ) +
  facet_wrap(
    vars(org, type),
    scales = "free"
  ) +
  coord_flip() +
  labs(x = "Topic", y = "Average Probability per Document")

## display plot
lda_summ_topics_plot
```




```r
### assign topics to documents
topic_class <- lda_topic_prob %>%
  group_by(id, org, type) %>%
  slice_max(gamma) %>%
  ungroup()

## print
topic_class
```

```
## # A tibble: 1,961 x 5
##    id    org    type  topic gamma
##    <chr> <chr>  <chr> <fct> <dbl>
##  1 1     Amazon Cons  6     0.834
##  2 1     Amazon Pros  3     0.451
##  3 10    Amazon Cons  3     0.587
##  4 10    Amazon Pros  6     0.542
##  5 100   Amazon Cons  2     0.393
##  6 100   Amazon Pros  4     0.425
##  7 1000  Google Cons  6     0.727
##  8 1000  Google Pros  7     0.363
##  9 1001  Google Cons  1     0.315
## 10 1001  Google Pros  2     0.392
## # ... with 1,951 more rows
```

Task 10: Evaluate Topic Model

For this task, you will evaluate the topic model.

Task 10.1

Create a new data table named **org_type_topic_class**. Pipe **topic_class** into **count()**. Count by **org**, **type**, and **topic**. Name the count as **comment_count**. Group by **org** and **type**. Slice for the top **comment_count** values. Remove the groups. Select **org** and rename it **org_top**, select **type** and rename it **type_top**, and select **topic**.

Pipe **topic_class** into **inner_join()** with **org_type_topic_class** and join by **topic**. Create a column named **match** to indicate whether **org** equals **org_top** and **type** equals **type_top**. Group by **id**, **org**, and **type**. Summarize by computing **match** via **sum()** applied to **match**. Drop the groups. Summarize again by computing **prop_match** via **sum()** applied to **match** and divided by the row count via **n()**. Drop the groups.

Question 10.1: What is the matching proportion of topics?

Response 10.1: *0.413*.

```
### compute most common topic per comment
org_type_topic_class <- topic_class %>%
  count(org, type, topic, name = "comment_count") %>%
  group_by(org, type) %>%
  slice_max(
    comment_count,
    n = 1
  ) %>%
  ungroup() %>%
  select(
    org_top = org,
    type_top = type,
    topic
  )

### correcting top classifications
topic_class %>%
  inner_join(
    org_type_topic_class,
    by = "topic"
  ) %>%
  mutate(
    match = case_when(
      org == org_top & type == type_top ~ 1,
      TRUE ~ 0
    )
  ) %>%
  group_by(id, org, type) %>%
  summarize(
    match = sum(match),
    .groups = "drop"
  ) %>%
  summarize(
    match_prop = sum(match) / n(),
```

```
    .groups = "drop"
  )
```

```
## # A tibble: 1 x 1
##   match_prop
##        <dbl>
## 1      0.413
```

Task 10.2

Create a data table named **org_type_topic_word_class**. Pipe **word_class** into **separate()** and separate the **document** column into **id**, **org**, and **type** columns. Pipe the result into **inner_join()** to join with **org_type_topic_class** by **topic**. Pipe the result into a *first* **unite()** to cnite **org** and **type** into **org_type**. Pipe the result into a *second* **unite()** to unite **org_top** and **type_top** into **org_type_top**.

Create a data table named **org_type_topic_word_class_summ**. Pipe **org_type_topic_word_class** into **count()** and count by **org_type** and **org_type_top** weighted by **count** naming the result **match**. Group by **org_type**. Calculate **match_prop** from **match** and the *sum* of **match**. Remove the groups.

Create a heat map named **org_type_topic_word_class_summ_plot**. Call **ggplot()**, set the *data* to **org_type_topic_word_class_summ**, map **org_type_top** to the *x-axis*, **org_type** to the *y-axis*, and **match_prop** to the *fill*. Add a **geom_tile()** layer. Scale the fill with **scale_fill_gradient2()** setting **low** to **blue**, **high** to **red**, **midpoint** to **0.25**, and **label** to **scales::percent_format()**. Choose the *minimal* theme. Label the axes and fill appropriately. Display the plot.

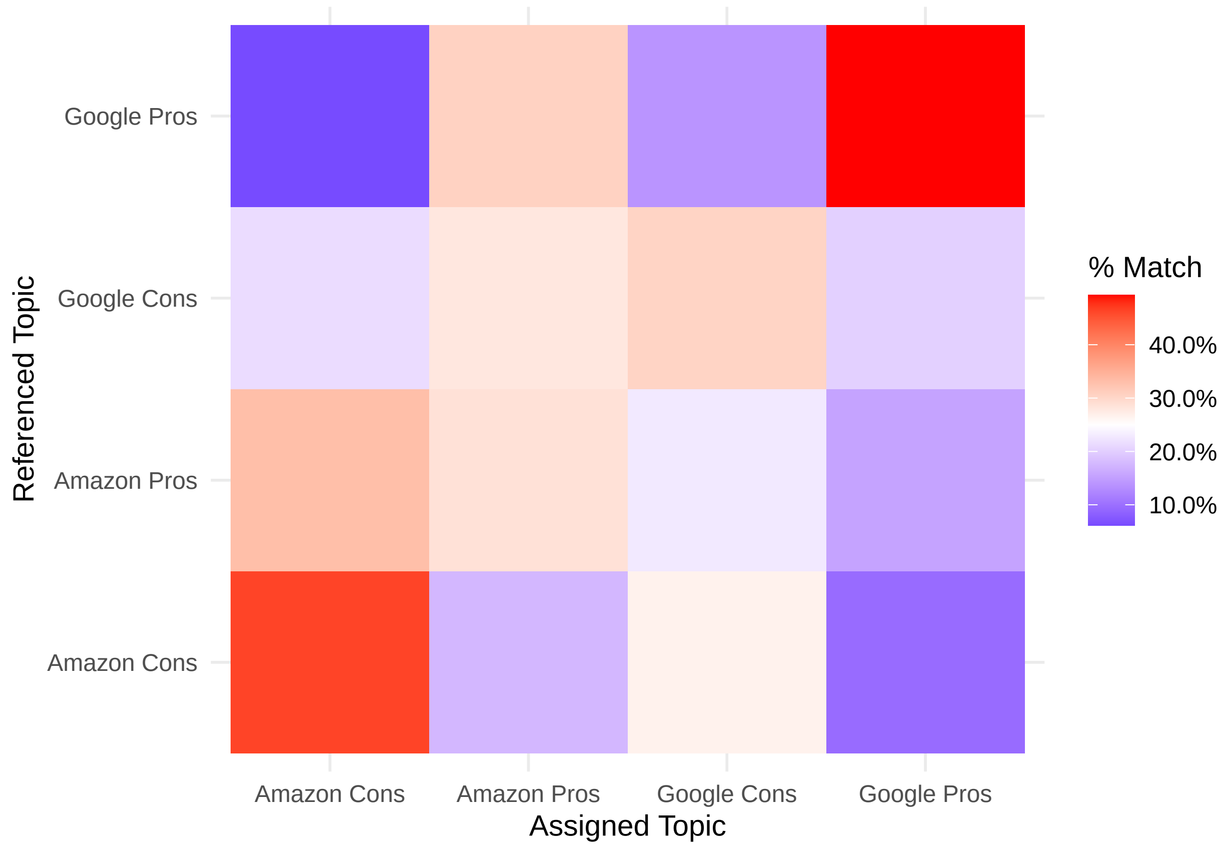
Pipe **org_type_topic_word_class** into **filter()** to filter for rows where **org_type** does *not* equal **org_type_top**. Count by **org_type**, **org_type_top**, and **term** weighted by **count**, naming the result **mismatch_count**. Remove the groups. Arrange by *descending* **mismatch_count**.

Questions 10.2: Answer these questions: (1) Which *two* combinations of *organization* and *comment type* have the *highest percentage topic match*? (2) Which *word* has the *highest mismatch count*?

Responses 10.2: *(1) The Google pros (2) people.*

```
### calculate word topic assignments
org_type_topic_word_class <- word_class %>%
  separate(
    document,
    c("id", "org", "type"),
    sep = " "
  ) %>%
  inner_join(
    org_type_topic_class,
    by = "topic"
  ) %>%
  unite(
    org_type,
    org, type,
    sep = " "
  ) %>%
  unite(
    org_type_top,
    org_top, type_top,
    sep = " "
  )
```

```r
### summary of word topic assignments
org_type_topic_word_class_summ <- org_type_topic_word_class %>%
  count(org_type, org_type_top, wt = count, name = "match") %>%
  group_by(org_type) %>%
  mutate(
    match_prop = match / sum(match)
  ) %>%
  ungroup()

### plot confusion matrix
org_type_topic_word_class_summ_plot <- ggplot(
  org_type_topic_word_class_summ,
  aes(
    x = org_type_top,
    y = org_type,
    fill = match_prop
  )
) +
  geom_tile() +
  scale_fill_gradient2(
    low = "blue",
    high = "red",
    midpoint = 0.25,
    label = scales::percent_format()
  ) +
  theme_minimal() +
  labs(
    x = "Assigned Topic",
    y = "Referenced Topic",
    fill = "% Match"
  )

## display plot
org_type_topic_word_class_summ_plot
```




```
### find incorrectly classified words
org_type_topic_word_class %>%
  filter(org_type != org_type_top) %>%
  count(org_type, org_type_top, term, wt = count, name = "mismatch_count") %>%
  ungroup() %>%
  arrange(desc(mismatch_count))
```

```
## # A tibble: 1,874 x 4
##    org_type    org_type_top term        mismatch_count
##    <chr>       <chr>        <chr>                <dbl>
##  1 Google Pros Amazon Pros  People                 141
##  2 Google Cons Amazon Pros  Company                 95
##  3 Amazon Pros Google Cons  Pay                     92
##  4 Google Pros Google Cons  Benefits                87
##  5 Amazon Pros Google Cons  Benefits                86
##  6 Google Pros Amazon Pros  Company                 64
##  7 Google Pros Amazon Pros  Smart                   63
##  8 Amazon Cons Amazon Pros  People                  59
##  9 Google Pros Amazon Pros  Environment             53
## 10 Google Cons Amazon Pros  People                  50
## # ... with 1,864 more rows
```

Task 11: Save Plots and Data

For this task, you will save created objects.

Task 11.1

Save **emp_reviews** as the data file **emp_reviews.txt** in the **data** folder of the project directory using **write_delim()**.

Save the ten plot objects as **png** files in the **plots** folder of the project directory. Use a width of *9 inches* and height of *9 inches* for all plots.

```
### save working data
write_delim(
  emp_reviews,
  file = here("data", "emp_reviews.txt"),
  delim = "|"
)

### save plots to folder in project directory
ggsave(
  here("plots", "afinn_sent.png"),
  plot = afinn_sent_plot,
  units = "in", width = 9, height = 9
)

## save a single plot to a file
ggsave(
  here("plots", "bigram_tg.png"),
  plot = bigram_tg_plot,
  units = "in", width = 9, height = 9
)

## save a single plot to a file
ggsave(
  here("plots", "bing_sent.png"),
  plot = bing_sent_plot,
  units = "in", width = 9, height = 9
)

## save a single plot to a file
ggsave(
  here("plots", "lda_summ_topics.png"),
  plot = lda_summ_topics_plot,
  units = "in", width = 9, height = 9
)

## save a single plot to a file
ggsave(
  here("plots", "lda_top_terms.png"),
  plot = lda_top_terms_plot,
  units = "in", width = 9, height = 9
)

## save a single plot to a file
ggsave(
  here("plots", "nrc_sent.png"),
  plot = nrc_sent_plot,
  units = "in", width = 9, height = 9
```

```
)

## save a single plot to a file
ggsave(
  here("plots", "org_type_topic_word_class_summ.png"),
  plot = org_type_topic_word_class_summ_plot,
  units = "in", width = 9, height = 9
)

## save a single plot to a file
ggsave(
  here("plots", "top_bigram_count.png"),
  plot = top_bigram_count_plot,
  units = "in", width = 9, height = 9
)

## save a single plot to a file
ggsave(
  here("plots", "word_cors_tg.png"),
  plot = word_cors_tg_plot,
  units = "in", width = 9, height = 9
)

## save a single plot to a file
ggsave(
  here("plots", "word_pairs_tg.png"),
  plot = word_pairs_tg_plot,
  units = "in", width = 9, height = 9
)
```

Task 12: Conceptual Questions

For your last task, you will respond to conceptual questions based on the conceptual lectures for this week.

Question 12.1: What does it mean to *tokenize* text? Provide examples.

Response 12.1: *Tokenizing text is essentially splitting text into smaller units such as indvidual words. These indivudal words are called tokens. For example, "I have a pet dog." and "I have a pet dog and a pet cat" Tokenzing the sentence would be splitting each word up in each the sentences.You can then run analysis and see that the word "dog" has a frequency of 2..*

Question 12.2: How is *sentiment analysis* performed?

Response 12.2: *Sentiment analysis classifies text by either positive, negative or neutral sentiment. You preform a sentiment analysis by joing tokens with a sentiment dictionary..*

Question 12.3: How does *latent Dirichlet allocation* of tokens work?

Response 12.3: *LDA is topic modeling that spots relationships between words in a group. First the number of words in the document are determined, Next, a topic mixture for the document is fixed over a set of chosen topics. Next, a topic is selected. Lastly, a word is picked based on the topics multinomial distribution. .*