Text mining with Amazon and Google

Introduction

The objective of this project was analyzing the data of the Amazon and Google employee reviews to interpret which company has a better work-life balance and percieved pay according to online reviews.

Seperating pros and cons

The data present in the datasets 'amzn' and 'goog' is anonymous online reviews collected from Glassdoor. Both datasets contains two columns 'pros' and 'cons' which need to be seperated for text analysis.

```
# Print the structure of amzn
str(amzn)
```

```
## 'data.frame':    500 obs. of  5 variables:
##  $ sr_no : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ pg_num: Factor w/ 59 levels "10","11","12",..: 44 44 44 44 44 44 44 44 44 44 ...
##  $ url   : Factor w/ 59 levels "https://www.glassdoor.com/Reviews/Amazon-com-Reviews-E6036_P10.htm",
##  $ pros  : Factor w/ 497 levels "4 day work week. Good pay and benefits. Energetic atmosphere. Abili
##  $ cons  : Factor w/ 496 levels "\"10 hour shifts on your feet, half hour lunch\"",..: 150 276 253 8
```

```
# Create amzn_pros
amzn_pros = amzn$pros

# Create amzn_cons
amzn_cons = amzn$cons

# Print the structure of goog
str(goog)
```

```
## 'data.frame':    501 obs. of  5 variables:
##  $ sr_no  : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ pg_num.: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ url    : Factor w/ 50 levels "https://www.glassdoor.com/Reviews/Google-Reviews-E9079_P10.htm",..:
##  $ pros   : Factor w/ 492 levels "(1) Countless perks that probably adds 20% to your base salary (in
##  $ cons   : Factor w/ 492 levels "1) Inexperienced and bloated middle management. 2) The white badge
```

```
# Create goog_pros
goog_pros = goog$pros

# Create goog_cons
goog_cons = goog$cons
```

Text organization

I'll be using the two functions to clean the data. qdap_clean(), which applies a series of qdap functions to a text vector, and tm_clean(), which applies a series of tm functions to a corpus object.

```
# qdap cleaning function
qdap_clean <- function(x) {
x <- replace_abbreviation(x)
x <- replace_contraction(x)
x <- replace_number(x)
x <- replace_ordinal(x)
x <- replace_symbol(x)
x <- tolower(x)
return(x)
}

tm_clean <- function(corpus) {
tm_clean <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords,
 c(stopwords("en"), "Google", "Amazon", "company"))
return(corpus)
}
```

Cleaning Amazon 'pros' and 'cons' data

```
# Alter amzn_pros
amzn_pros = qdap_clean(amzn_pros)

# Alter amzn_cons
amzn_cons = qdap_clean(amzn_cons)

# Create az_p_corp
az_p_corp = VCorpus(VectorSource(amzn_pros))

# Create az_c_corp
az_c_corp = VCorpus(VectorSource(amzn_cons))

# Create amzn_pros_corp
amzn_pros_corp = tm_clean(az_p_corp)

# Create amzn_cons_corp
amzn_cons_corp = tm_clean(az_c_corp)
```

Cleaning Google 'pros' and 'cons' data

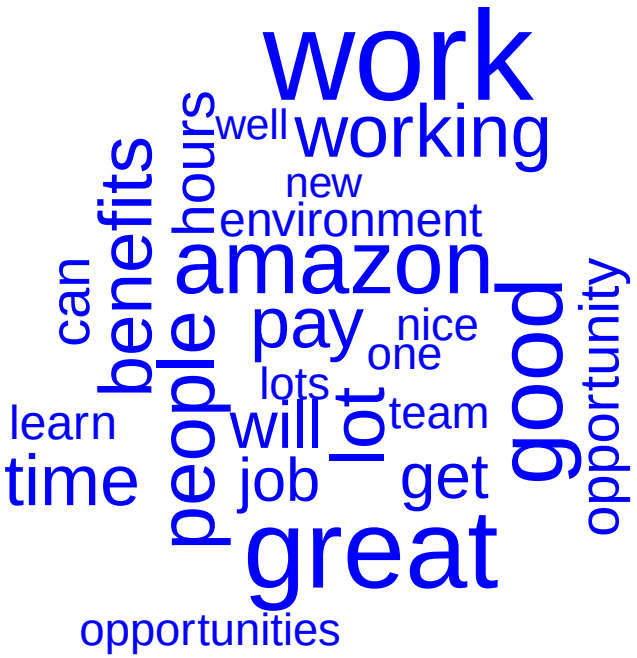
```
# Apply qdap_clean to goog_pros
goog_pros = qdap_clean(goog_pros)

# Apply qdap_clean to goog_cons
goog_cons = qdap_clean(goog_cons)

# Create goog_p_corp
goog_p_corp = VCorpus(VectorSource(goog_pros))
```

```
# Create goog_c_corp
goog_c_corp = VCorpus(VectorSource(goog_cons))

# Create goog_pros_corp
goog_pros_corp = tm_clean(goog_p_corp)

# Create goog_cons_corp
goog_cons_corp = tm_clean(goog_c_corp)
```

Feature extraction & analysis : Amazon pros

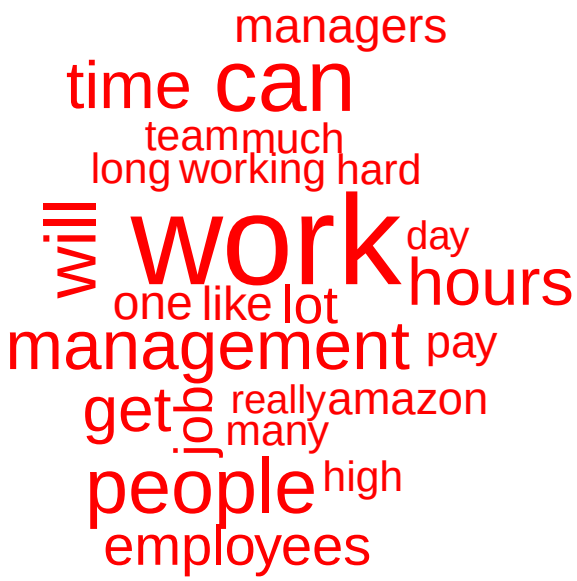
amzn_pros_corp, amzn_cons_corp, goog_pros_corp and goog_cons_corp have all been preprocessed, so now we can extract the features we want to examine. We will be using bag of words approch for that we need to create a bigram TermDocumentMatrix for Amazon's positive reviews corpus, amzn_pros_corp. From this, we can quickly create a wordcloud() to understand what phrases people positively associate with working at Amazon.

```
# Make tokenizer function
tokenizer = function(x)
   NGramTokenizer(x , Weka_control(min =2 ,max =2))

# Create amzn_p_tdm
amzn_p_tdm = TermDocumentMatrix(amzn_pros_corp , control = list(tokenize = tokenizer))

# Create amzn_p_tdm_m
amzn_p_tdm_m = as.matrix(amzn_p_tdm)

# Create amzn_p_freq
amzn_p_freq = rowSums(amzn_p_tdm_m)

# Plot a wordcloud using amzn_p_freq values
wordcloud(names(amzn_p_freq) , max.words =25 , color = "blue")
```



Feature extraction & analysis : Amazon cons

```
# Create amzn_c_tdm
amzn_c_tdm = TermDocumentMatrix(amzn_cons_corp , control = list(tokenize =tokenizer))

# Create amzn_c_tdm_m
amzn_c_tdm_m = as.matrix(amzn_c_tdm)

# Create amzn_c_freq
amzn_c_freq = rowSums(amzn_c_tdm_m)

# Plot a wordcloud of negative Amazon bigrams
wordcloud(names(amzn_c_freq) , max.words = 25 , color = "red")
```



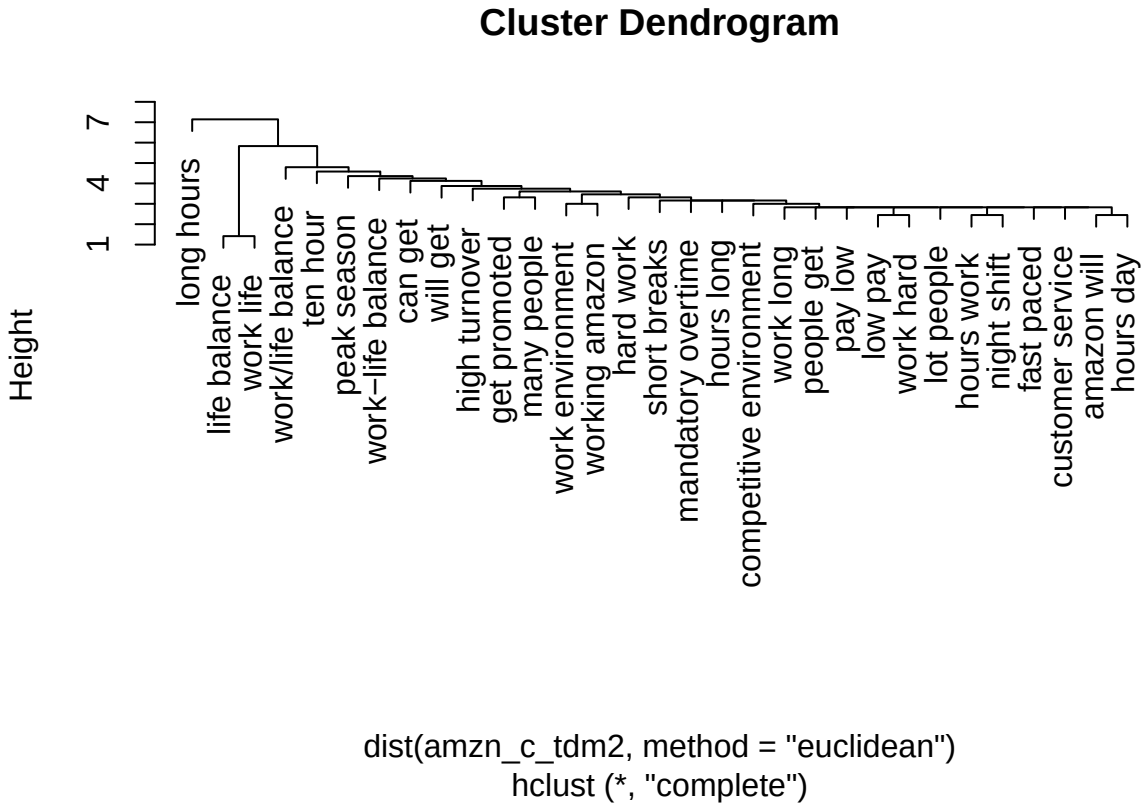
Feature extraction & analysis : Amazon cons dendogram

There is a strong indication of long working hours and poor work-life balance in the reviews. A simple clustering technique, hierarchical cluster and a dendrogram can be used to see how connected these phrases are.

```
# Create amzn_c_tdm
amzn_c_tdm = TermDocumentMatrix(amzn_cons_corp , control = list(tokenize = tokenizer))

# Print amzn_c_tdm to the console
amzn_c_tdm
```

```
## <<TermDocumentMatrix (terms: 4836, documents: 500)>>
## Non-/sparse entries: 5283/2412717
## Sparsity           : 100%
## Maximal term length: 32
## Weighting          : term frequency (tf)
```

```
# Create amzn_c_tdm2 by removing sparse terms
amzn_c_tdm2 = removeSparseTerms(amzn_c_tdm , sparse = 0.993)

# Create hc as a cluster of distance values
hc = hclust(dist(amzn_c_tdm2 ,method = "euclidean") , method ="complete" )

# Produce a plot of hc
plot(hc)
```



Feature extraction & analysis : Word Association

As expected, we can see similar topics throughout the dendrogram. Switching back to positive comments, we need to examine top phrases that appeared in the word clouds. We need to find associated terms using the findAssocs() function from tm.

```
# Create amzn_p_tdm
amzn_p_tdm = TermDocumentMatrix(amzn_pros_corp , control = list(tokenize = tokenizer))

# Create amzn_p_m
amzn_p_m = as.matrix(amzn_p_tdm)

# Create amzn_p_freq
amzn_p_freq = rowSums(amzn_p_m)

# Create term_frequency
term_frequency = sort(amzn_p_freq , decreasing = TRUE)

# Print the 5 most common terms
term_frequency[1:5]
```

```
##       good pay great benefits   smart people     place work     fast paced
##             25             24             20             17             16
```

```
# Find associations with fast paced
findAssocs(amzn_p_tdm , "fast paced" , 0.2)
```

```
## $`fast paced`
##        paced environment       environments ever               learn fast
##                     0.49                    0.35                     0.35
##           paced friendly             paced work              able excel
##                     0.35                    0.35                     0.25
##           activity ample            advance one               also well
##                     0.25                    0.25                     0.25
##              amazon fast          amazon noting              amazon one
##                     0.25                    0.25                     0.25
##              amount time       ample opportunity       assistance ninety
##                     0.25                    0.25                     0.25
##           break computer          call activity              can choose
##                     0.25                    0.25                     0.25
##            catchy cheers          center things      challenging expect
##                     0.25                    0.25                     0.25
##       cheers opportunity         choose success  combined encouragement
##                     0.25                    0.25                     0.25
## competitive environments         computer room             cool things
##                     0.25                    0.25                     0.25
##          deliver results            dock makes          driven deliver
##                     0.25                    0.25                     0.25
##               easy learn       emphasis shipping encouragement innovation
##                     0.25                    0.25                     0.25
##     environment benefits      environment catchy      environment center
##                     0.25                    0.25                     0.25
##         environment fast       environment help       environment smart
##                     0.25                    0.25                     0.25
##       ever-changing fast            ever known          ever witnessed
##                     0.25                    0.25                     0.25
##               everyone s          excel advance     excel ever-changing
##                     0.25                    0.25                     0.25
##     exciting environment          expect learn          extremely fast
##                     0.25                    0.25                     0.25
##             facility top        fail successful          fantastic able
##                     0.25                    0.25                     0.25
##               fired part           five percent          freindly place
##                     0.25                    0.25                     0.25
##      friendly atmosphere     friendly management            full medical
##                     0.25                    0.25                     0.25
##                get fired          go extremely            great plenty
##                     0.25                    0.25                     0.25
##           great teamwork    happening technology         hassle benefits
##                     0.25                    0.25                     0.25
##                 help get          help workers            high quality
##                     0.25                    0.25                     0.25
##              high volume         including full       innovation owning
##                     0.25                    0.25                     0.25
##         job requirements            leader can              line break
##                     0.25                    0.25                     0.25
##       lot responsibility       maintaining high             makes time
##                     0.25                    0.25                     0.25
##          management nice          nice facility            ninety five
##                     0.25                    0.25                     0.25
##             noting short      offers opportunity         one competitive
```

```
##                  0.25                   0.25                  0.25
##              one fast   opportunity overtime       opportunity yell
##                  0.25                   0.25                  0.25
##        ownership fast            owning work               paced -
##                  0.25                   0.25                  0.25
##        paced emphasis         paced exciting            paced high
##                  0.25                   0.25                  0.25
##           paced never         paced rewarding            paced ship
##                  0.25                   0.25                  0.25
##        paced software           paid upfront        people focused
##                  0.25                   0.25                  0.25
##          percent paid          plenty shifts          position fast
##                  0.25                   0.25                  0.25
##        possible still       preferences fast       products quickly
##                  0.25                   0.25                  0.25
##           quality bar       quickly possible      readily available
##                  0.25                   0.25                  0.25
##      requirements easy responsibility ownership         results great
##                  0.25                   0.25                  0.25
##          results team       rewarding people        shifts everyone
##                  0.25                   0.25                  0.25
##             ship dock      shipping products           short amount
##                  0.25                   0.25                  0.25
##       short fantastic       smart co-workers         s preferences
##                  0.25                   0.25                  0.25
##      still maintaining           success fail        successful also
##                  0.25                   0.25                  0.25
##           team driven       technology today       things happening
##                  0.25                   0.25                  0.25
##            things lot              time fast                 time go
##                  0.25                   0.25                  0.25
##             top line     upfront experience           vision well
##                  0.25                   0.25                  0.25
##           volume call          well rewarded           well tuition
##                  0.25                   0.25                  0.25
##      witnessed combined              work can              work cool
##                  0.25                   0.25                  0.25
##       work environments        workers readily              work fast
##                  0.25                   0.25                  0.25
##             work job            yell leader
##                  0.25                   0.25
```

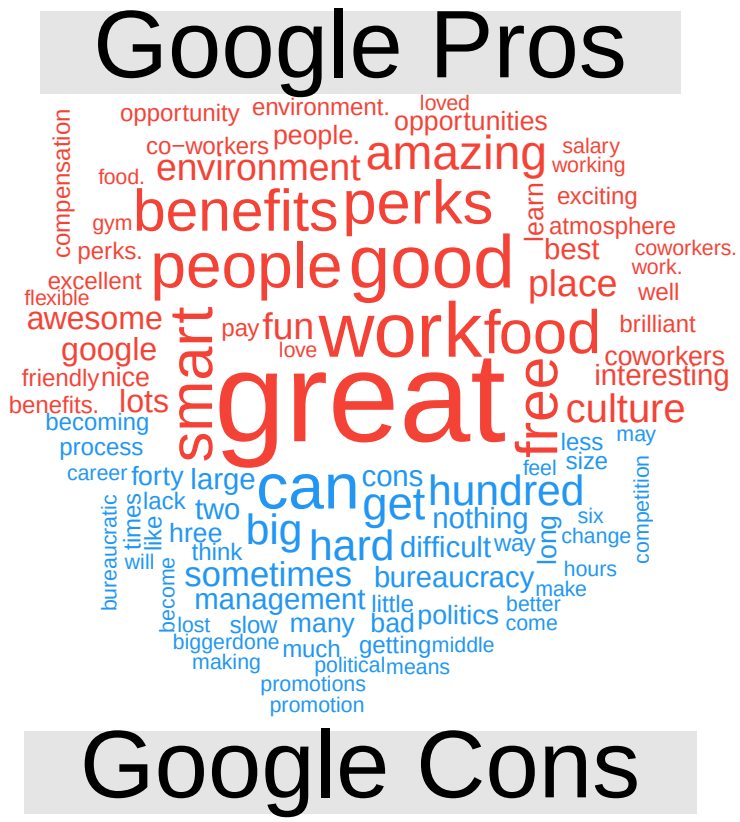
Reviewing the associated terms reveals that some are still somewhat negative.

Quick review of Google

We need to create a comparison.cloud() of Google's positive and negative reviews for comparison to Amazon. This will give us a quick understanding of top terms without having to spend as much time as we did examining the Amazon reviews in the previous exercises.

The all_goog_corpus, which has the 500 positive and 500 negative reviews for Google. Here we need to clean the corpus and create a comparison cloud comparing the common words in both pro and con reviews.

```
# Create total_goog
total_goog = rbind(goog_pros , goog_cons)

# Create goog_source
goog_source = DataframeSource(total_goog)

# Create all_goog
all_goog = VCorpus(goog_source)

# Clean all_goog_corp
all_goog_corp = tm_clean(all_goog)

# Create all_tdm
all_tdm = TermDocumentMatrix(all_goog_corp)

# Name the tdm columns
colnames(all_tdm) = c("Google Pros" , "Google Cons")

# Create all_m
all_m = as.matrix(all_tdm)

# Build a comparison cloud
comparison.cloud(all_m , colors = c("#F44336", "#2196f3") , max.words = 100)
```



Amazon vs. Google pro reviews

Positive Amazon reviews appear to mention "good benefits" while the negative reviews focus on "work load" and "work-life balance" issues.

In contrast, Google's positive reviews mention "great food", "perks", "smart people", and "fun culture", among other things. The Google negative reviews discuss "politics", "getting big", "bureaucracy" and "middle management".

We need to make a pyramid plot lining up positive reviews for Amazon and Google so we can adequately see the differences between any shared birgrams.

```
# Structure of Amazon pros
str(amzn_pros)
```

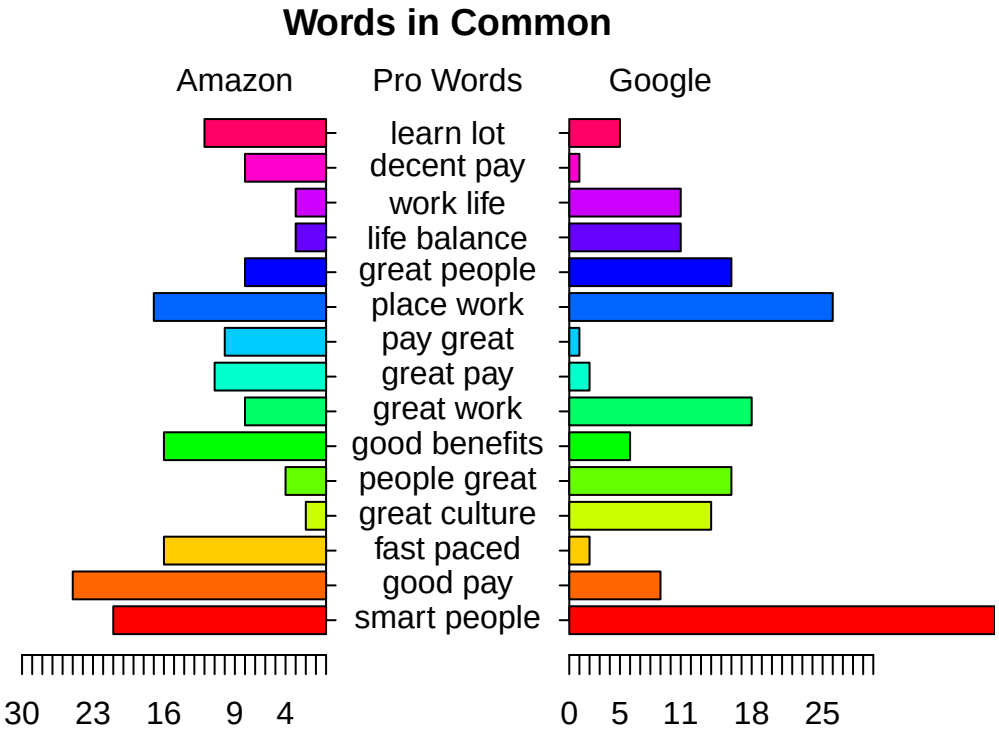
```
##  chr [1:500] "you are surrounded by smart people and the projects are interesting if a little dauntir
```

```
# Structure of Google pros
str(goog_pros)
```

```
##  chr [1:501] "* if you are a software engineer you are among the kings of the hill at google. it is a
```

```
# Create total_pros
total_pros = rbind(amzn_pros , goog_pros)

# Create pros_source
pros_source = DataframeSource(total_pros)

# Create all_pros
all_pros = VCorpus(pros_source)

# Create all_pros_corp
all_pros_corp = tm_clean(all_pros)

# Create bigram TDM
all_pros_tdm = TermDocumentMatrix(all_pros_corp , control = list(tokenize = tokenizer))

# Name the columns
colnames(all_pros_tdm) = c("Amazon Pros" , "Google Pros")

# Create all_tdm_pm
all_tdm_pm = as.matrix(all_pros_tdm)

# Create common_words
 common_pros_words <- subset(all_tdm_pm,
                         all_tdm_pm[, 1] > 0 & all_tdm_pm[, 2] > 0)

# Create difference
 difference = abs(common_pros_words[, 1] - common_pros_words[, 2])

# Add difference to common_pros_words
common_pros_words= cbind(common_pros_words, difference)
```

```
# Order the data frame from most differences to least
common_pros_words  = common_pros_words[order(common_pros_words[, 3],
                                  decreasing = TRUE), ]
# Create top15_df_pros
top15_df_pros = data.frame (x = common_pros_words[1:15, 1] ,
                        y = common_pros_words[1:15, 2] ,
                        labels = rownames(common_pros_words[1:15 , ]))

# Create the pyramid plot
pyramid.plot(top15_df_pros$x, top15_df_pros$y, labels  = top15_df_pros$labels,
             gap = 12, main = "Words in Common", unit = NULL,
              top.labels = c("Amazon", "Pro Words", "Google"))
```




```
## [1] 5.1 4.1 4.1 2.1
```

Amazon vs. Google con reviews

Interestingly, some Amazon employees discussed "work-life balance" as a positive. In both organizations, people mentioned "culture" and "smart people", so there are some similar positive aspects between the two companies.

We need to make a pyramid plot lining up negative reviews for Amazon and Google so we can adequately see the differences between any shared birgrams.

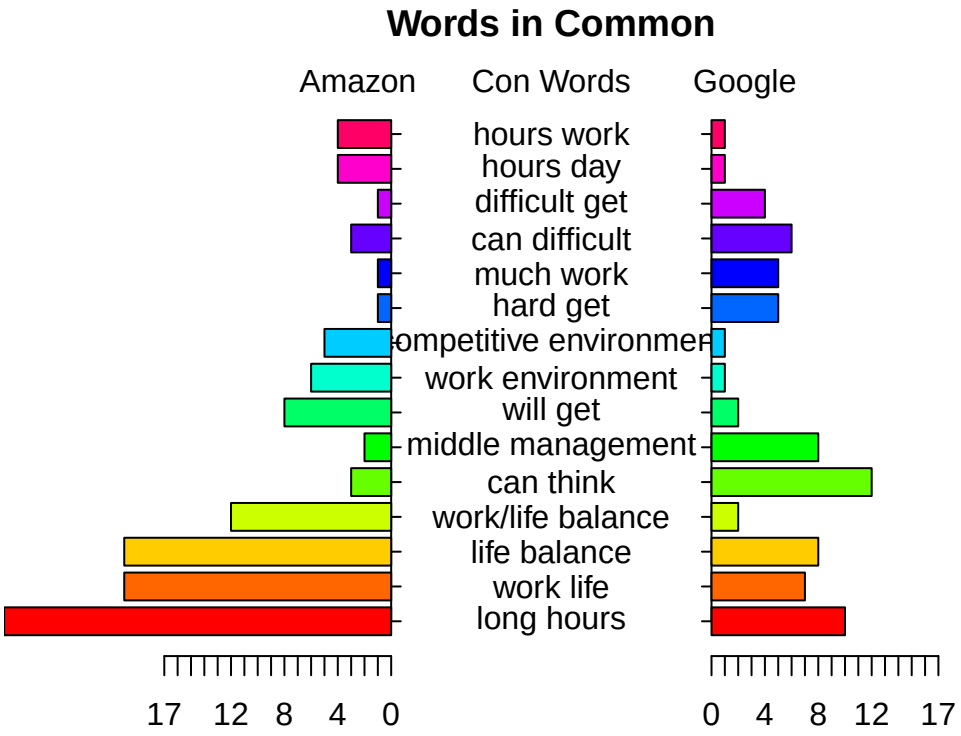
```
# Structure of Amazon cons
str(amzn_cons)
```

```
##  chr [1:500] "internal tools proliferation has created a mess for trying to get to basic information
```

```r
#Structure of Google cons
str(goog_cons)
```

```
##  chr [1:501] "* it *is* becoming larger and with it comes growing pains: bureaucracy slow to respond
```

```r
# Create total_cons
total_cons = rbind(amzn_cons , goog_cons)

# Create cons_source
cons_source = DataframeSource(total_cons)

# Create all_cons
all_cons = VCorpus(cons_source)

# Create all_cons_corp
all_cons_corp = tm_clean(all_cons)

# Create bigram TDM
all_cons_tdm = TermDocumentMatrix(all_cons_corp , control = list(tokenize = tokenizer))

# Name the columns
colnames(all_cons_tdm) = c("Amazon Cons" , "Google Cons")

# Create all_tdm_cm
all_tdm_cm = as.matrix(all_cons_tdm)


# Create common_cons_words
 common_cons_words <- subset(all_tdm_cm,
                         all_tdm_cm[, 1] > 0 & all_tdm_cm[, 2] > 0)

# Create difference
 difference = abs(common_cons_words[, 1] - common_cons_words[, 2])

# Add difference to common_cons_words
common_cons_words= cbind(common_cons_words, difference)

# Order the data frame from most differences to least
common_cons_words  = common_cons_words[order(common_cons_words[, 3],
                                  decreasing = TRUE), ]
# Create top15_df_cons
top15_df_cons = data.frame (x = common_cons_words[1:15, 1] ,
                        y = common_cons_words[1:15, 2] ,
                        labels = rownames(common_cons_words[1:15 , ]))

# Create the pyramid plot
pyramid.plot(top15_df_cons$x, top15_df_cons$y, labels  = top15_df_cons$labels,
              gap = 12, main = "Words in Common", unit = NULL,
               top.labels = c("Amazon", "Con Words", "Google"))
```




```
## [1] 5.1 4.1 4.1 2.1
```

Conclusion, insight and recommendation

Based on the visuals Google has a better work-life balance according to current employee reviews.

Earlier we had seen "fast paced" in the pros despite the other reviews mentioning "work-life balance". We will use findAssocs() to get a named vector of phrases. This may lead us to a conclusion about the type of person who favorably views an intense workload.

```
# Find Associations
findAssocs(amzn_p_tdm, "fast paced", 0.2)[[1]][1:15]
```

```
## paced environment environments ever        learn fast    paced friendly
##              0.49              0.35              0.35              0.35
##        paced work        able excel    activity ample       advance one
##              0.35              0.25              0.25              0.25
##         also well       amazon fast     amazon noting        amazon one
##              0.25              0.25              0.25              0.25
##       amount time ample opportunity assistance ninety
##              0.25              0.25              0.25
```

Given the abbreviated results of the associated phrases, we would recommend Amazon HR recruiters to identify candidates that view an intense workload as an opportunity to learn fast and give them ample opportunity.