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 colunms '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",..: 43 43 43 43 43 43 43 43 43 43 ...  
## $ pros : Factor w/ 497 levels "4 day work week. Good pay and benefits. Energetic atmosphere. Ability to see and familiarize yourself with many many products",..: 490 53 149 349 358 368 180 421 200 352 ...  
## $ cons : Factor w/ 496 levels "\"10 hour shifts on your feet, half hour lunch\"",..: 150 276 253 86 289 194 375 210 110 159 ...

# 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",..: 11 11 11 11 11 11 11 11 11 11 ...  
## $ pros : Factor w/ 492 levels "(1) Countless perks that probably adds 20% to your base salary (including free food transportation gym dance/sports classes etc"| \_\_truncated\_\_,..: 291 2 486 488 404 222 407 368 299 376 ...  
## $ cons : Factor w/ 492 levels "1) Inexperienced and bloated middle management. 2) The white badge vs red badge situation creates conflict within departments. "| \_\_truncated\_\_,..: 191 7 167 121 291 44 451 445 168 98 ...

# 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)

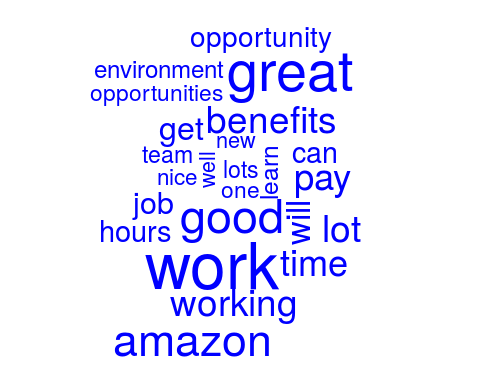
### Cleaning Google 'pros' and 'cons' data

# 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

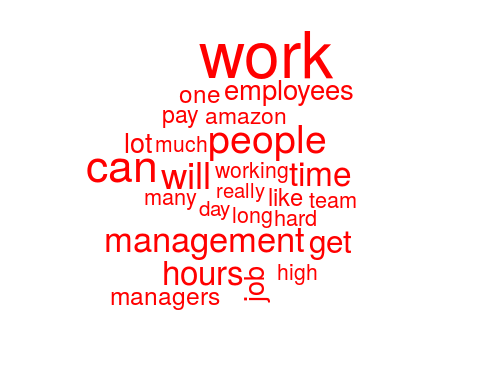
amzn\_pros\_corp, amzn\_cons\_corp, goog\_pros\_corp and goog\_cons\_corp have all been preprocessed, so now we can extract the features we want to examine. We will be using bag of words approch for that we need to create a bigram TermDocumentMatrix for Amazon's positive reviews corpus, amzn\_pros\_corp. 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")



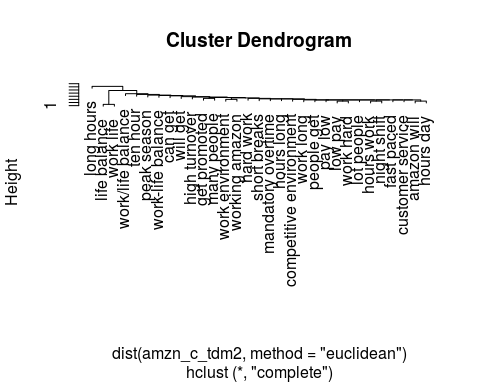
### Feature extraction & analysis : Amazon cons dendogram

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.

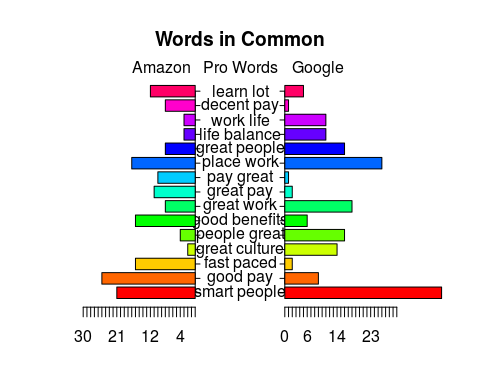
# Structure of Amazon pros  
str(amzn\_pros)

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

# 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 an engineer-driven company without a doub"| \_\_truncated\_\_ ...

# 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.

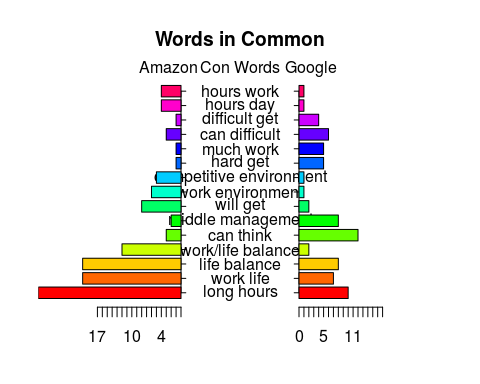
# Structure of Amazon cons  
str(amzn\_cons)

## chr [1:500] "internal tools proliferation has created a mess for trying to get to basic information. most people are required to learn/under"| \_\_truncated\_\_ ...

#Structure of Google cons  
str(goog\_cons)

## chr [1:501] "\* it \*is\* becoming larger and with it comes growing pains: bureaucracy slow to respond to market threats bloated teams cross-di"| \_\_truncated\_\_ ...

# 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.