A case study in HR analytics

Shubham

Steps for text mining project

- **Problem Defination and Specific Goal**
- **Identify Text to be collected**
- **Text Organization**
- **Feature Extraction**
- **Analysis**
- **Reach an insight**

Problem :- Which company has better work life balance? Which has better perceived pay according to online reviews?We learn something about how employees review both Amazon and Google.

Employee reviews can come from various sources.Forbes and others publish articles about the "best places to work", which may mention Amazon and Google. Another source of information might be anonymous online reviews from websites like Indeed, Glassdoor or CareerBliss.

Here, we'll focus on a collection of anonymous online reviews of amazon and google.

```
library(readr)
amazon <- read_csv("~/500_amzn.csv")
google <- read_csv("~/500_goog.csv")
```

```
str(amazon)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    500 obs. of  4 variables:
##  $ pg_num: int  50 50 50 50 50 50 50 50 50 50 ...
##  $ url   : chr  "https://www.glassdoor.com/Reviews/Amazon-com-Reviews-E6036_P50.htm" "https://www.gla
##  $ pros  : chr  "You're surrounded by smart people and the projects are interesting, if a little daun
##  $ cons  : chr  "Internal tools proliferation has created a mess for trying to get to basic informati
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 4
##   .. ..$ pg_num: list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ url   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ pros  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ cons  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

```
amazon_pros <- amazon$pros
amazon_cons <- amazon$cons
```

```
str(google)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    501 obs. of  4 variables:
##  $ pg_num: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ url   : chr  "https://www.glassdoor.com/Reviews/Google-Reviews-E9079_P1.htm" "https://www.glassdoo
##  $ pros  : chr  "* If you're a software engineer, you're among the kings of the hill at Google. It's
```

```
##  $ cons  : chr  "* It *is* becoming larger, and with it comes growing pains: bureaucracy, slow to re
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 4
##   .. ..$ pg_num: list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ url   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ pros  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ cons  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

```
google_pros <- google$pros
google_cons <- google$cons
```

Text organization Now that we have selected the exact text sources, we are now ready to clean them up. We'll be using the two functions 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.

```
library(qdap)
qdap_clean <- function(x){
  x<- na.omit(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)
}
```

```
library(tm)
```

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

Applying qdap_clean() to amazon and google

```
amazon_pros <- qdap_clean(amazon_pros)
amazon_cons <- qdap_clean(amazon_cons)

google_pros <- qdap_clean(google_pros)
google_cons <- qdap_clean(google_cons)
```

Next step is to convert this vector containing the text data to a corpus. Corpus is a collection of documents, but it's also important to know that in the tm domain, R recognizes it as a data type.

There are two kinds of the corpus data type, the permanent corpus, PCorpus, and the volatile corpus, VCorpus. In essence, the difference between the two has to do with how the collection of documents is stored in your computer.We will use the volatile corpus, which is held in computer's RAM rather than saved to disk, just to be more memory efficient.

To make a volatile corpus, R needs to interpret each element in our vector of text, amazon_pros, as a document. And the tm package provides what are called Source functions to do just that! We'll use a Source function called VectorSource() because our text data is contained in a vector. The output of this function is called a Source.

```
amazon_p_corp <- VCorpus(VectorSource(amazon_pros))
amazon_c_corp <- VCorpus(VectorSource(amazon_cons))

google_p_corp <- VCorpus(VectorSource(google_pros))
google_c_corp <- VCorpus(VectorSource(google_cons))
```

Now using tm_clean to clean data

```
amazon_pros_corp <- tm_clean(amazon_p_corp)
amazon_cons_corp <- tm_clean(amazon_c_corp)

google_pros_corp <- tm_clean(google_p_corp)
google_cons_corp <- tm_clean(google_c_corp)
```

Steps 4 & 5: Feature extraction & analysis

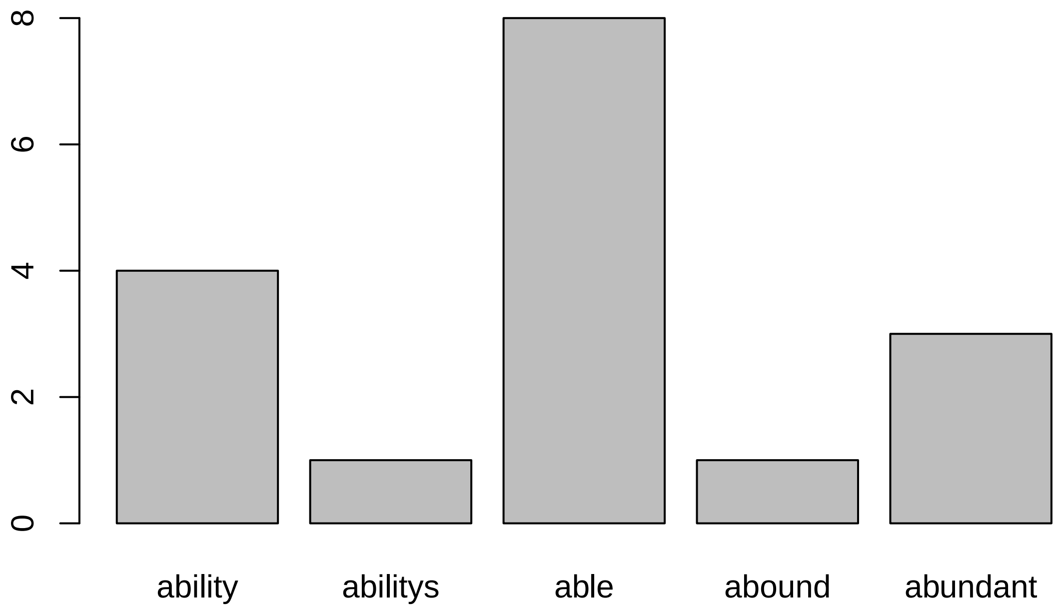
Since amzn_pros_corp, amzn_cons_corp, goog_pros_corp and goog_cons_corp have all been preprocessed, so now we can extract the features we want to examine. Since we are using the bag of words approach, we decide to create a bigram TermDocumentMatrix for Amazon's positive reviews corpus, amzn_pros_corp.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

The function below uses RWeka to tokenize two terms.

```
library(RWeka)
tokenizer <- function(x)
  NGramTokenizer(x, Weka_control(min = 2, max = 2))
```

Feature extraction & analysis: amazon_cons

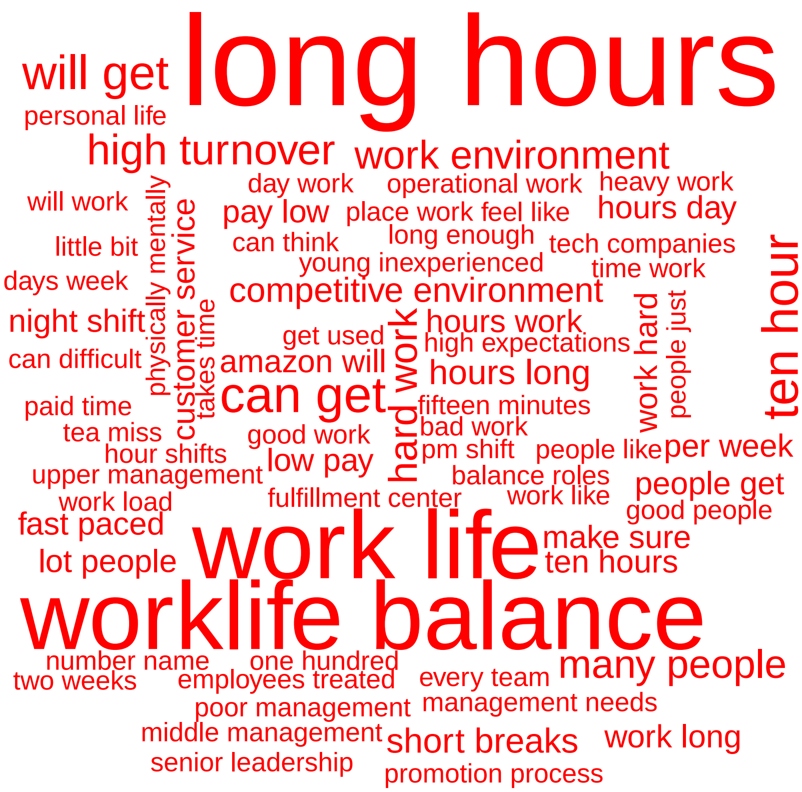
```
amazon_p_tdm <- TermDocumentMatrix(amazon_pros_corp)
amazon_p_tdm_m <- as.matrix(amazon_p_tdm)
amazon_p_freq <- rowSums(amazon_p_tdm_m)
amazon_p_f.sort <- sort(amazon_p_freq,decreasing = TRUE)

barplot(amazon_p_freq[1:5])
```




```
library(wordcloud)
amazon_p_tdm <- TermDocumentMatrix(amazon_pros_corp,control = list(tokenize=tokenizer))
amazon_p_tdm_m <- as.matrix(amazon_p_tdm)
amazon_p_freq <- rowSums(amazon_p_tdm_m)
amazon_p_f.sort <- sort(amazon_p_freq,decreasing = TRUE)
p_df <- data.frame(term=names(amazon_p_f.sort),num=amazon_p_f.sort)

wordcloud(p_df$term,p_df$num,max.words=100,color="red")
```

Feature extraction & analysis: amazon_cons

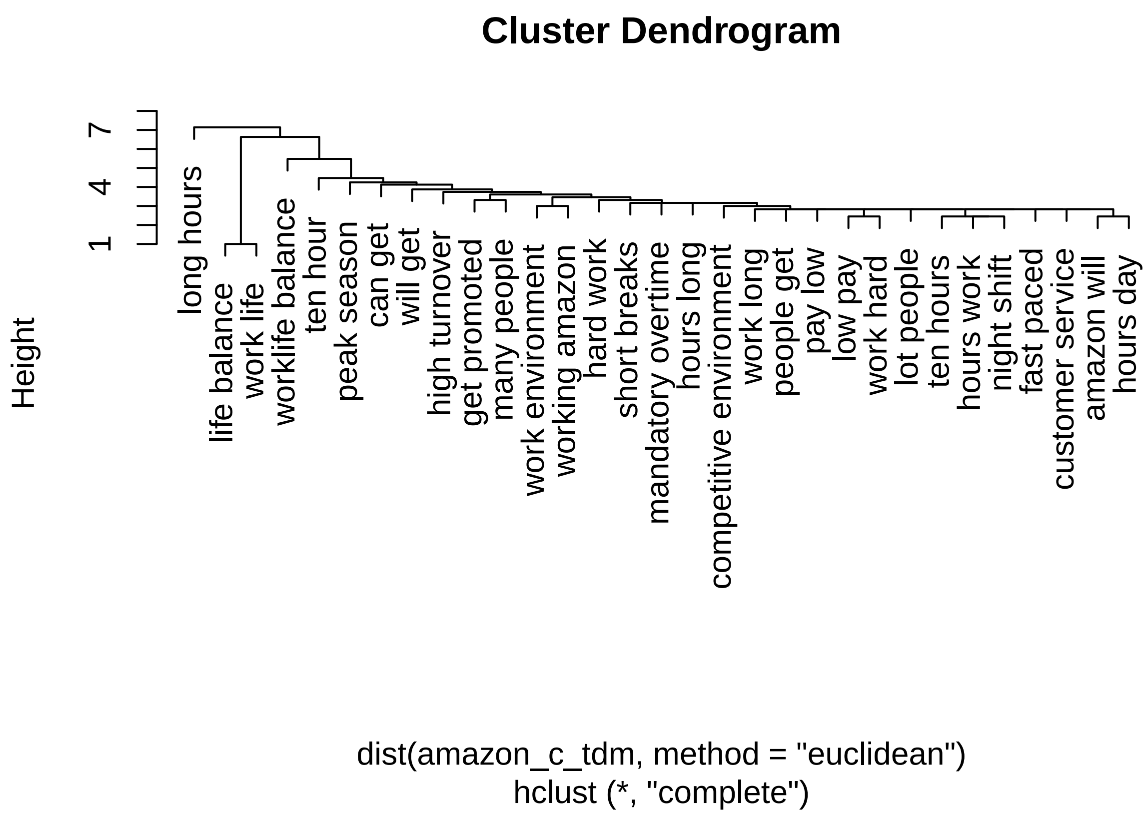
```
amazon_c_tdm <- TermDocumentMatrix(amazon_cons_corp,control=list(tokenize=tokenizer))
amazon_c_tdm_m <- as.matrix(amazon_c_tdm)
amazon_c_freq <- rowSums(amazon_c_tdm_m)
amazon_c_f.sort <- sort(amazon_c_freq,decreasing = TRUE)
c_df <- data.frame(term=names(amazon_c_f.sort),num=amazon_c_f.sort)

wordcloud(c_df$term,c_df$num,max.words=100,color="red")
```



amazon_cons dendrogram

It seems there is a strong indication of long working hours and poor work-life balance in the reviews. As a simple clustering technique, we'll decide to perform a hierarchical cluster and create a dendrogram to see how connected these phrases are.

```
amazon_c_tdm <- TermDocumentMatrix(amazon_cons_corp,control = list(tokenize=tokenizer))
amazon_c_tdm <- removeSparseTerms(amazon_c_tdm,0.993)

amazon_c_hclust <- hclust(dist(amazon_c_tdm,method="euclidean"),method="complete")

plot(amazon_c_hclust)
```



Word association Switching back to positive comments, we'll decide to examine top phrases that appeared in the word clouds. We'll now hope to find associated terms using the findAssocs() function from tm package.

```
amazon_p_tdm <- TermDocumentMatrix(amazon_pros_corp,control=list(tokenize=tokenizer))
amazon_p_m <- as.matrix(amazon_p_tdm)
amazon_p_freq <- rowSums(amazon_p_m)
token_frequency <- sort(amazon_p_freq,decreasing = TRUE)
token_frequency[1:5]
```

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

```
findAssocs(amazon_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
##      benefits including          break computer           call activity
##                    0.25                    0.25                    0.25
##              can choose          catchy company           center things
##                    0.25                    0.25                    0.25
##      challenging expect     cheers opportunity         choose success
```

```
##                  0.25                   0.25                   0.25
##  combined encouragement         company cheers competitive environments
##                  0.25                   0.25                   0.25
##           computer room            cool things        deliver results
##                  0.25                   0.25                   0.25
##              dock makes         driven deliver             easy learn
##                  0.25                   0.25                   0.25
##       emphasis shipping encouragement innovation    environment benefits
##                  0.25                   0.25                   0.25
##      environment catchy     environment center       environment fast
##                  0.25                   0.25                   0.25
##        environment help      environment smart       everchanging fast
##                  0.25                   0.25                   0.25
##              ever known         ever witnessed   everyones preferences
##                  0.25                   0.25                   0.25
##           excel advance     excel everchanging    exciting environment
##                  0.25                   0.25                   0.25
##            expect learn         extremely fast            facility top
##                  0.25                   0.25                   0.25
##         fail successful          fantastic able              fired part
##                  0.25                   0.25                   0.25
##            five percent         freindly place    friendly atmosphere
##                  0.25                   0.25                   0.25
##     friendly management           full medical               get fired
##                  0.25                   0.25                   0.25
##            go extremely           great plenty          great teamwork
##                  0.25                   0.25                   0.25
##    happening technology        hassle benefits                help get
##                  0.25                   0.25                   0.25
##            help workers           high quality             high volume
##                  0.25                   0.25                   0.25
##          including full      innovation owning        job requirements
##                  0.25                   0.25                   0.25
##              leader can             line break     lot responsibility
##                  0.25                   0.25                   0.25
##        maintaining high             makes time         management nice
##                  0.25                   0.25                   0.25
##           nice facility            ninety five            noting short
##                  0.25                   0.25                   0.25
##      offers opportunity        one competitive                one fast
##                  0.25                   0.25                   0.25
##    opportunity overtime       opportunity yell          ownership fast
##                  0.25                   0.25                   0.25
##              owning work         paced emphasis          paced exciting
##                  0.25                   0.25                   0.25
##              paced high            paced never         paced rewarding
##                  0.25                   0.25                   0.25
##              paced ship         paced software            paid upfront
##                  0.25                   0.25                   0.25
##          people focused           percent paid           plenty shifts
##                  0.25                   0.25                   0.25
##           position fast         possible still        preferences fast
##                  0.25                   0.25                   0.25
##        products quickly            quality bar        quickly possible
```

```
##                   0.25                    0.25                    0.25
##       readily available       requirements easy responsibility ownership
##                   0.25                    0.25                    0.25
##           results great            results team         rewarding people
##                   0.25                    0.25                    0.25
##         shifts everyones               ship dock        shipping products
##                   0.25                    0.25                    0.25
##            short amount         short fantastic          smart coworkers
##                   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
```

We decide to create a comparison.cloud() of Google's positive and negative reviews for comparison to Amazon. This will give you a quick understanding of top terms.

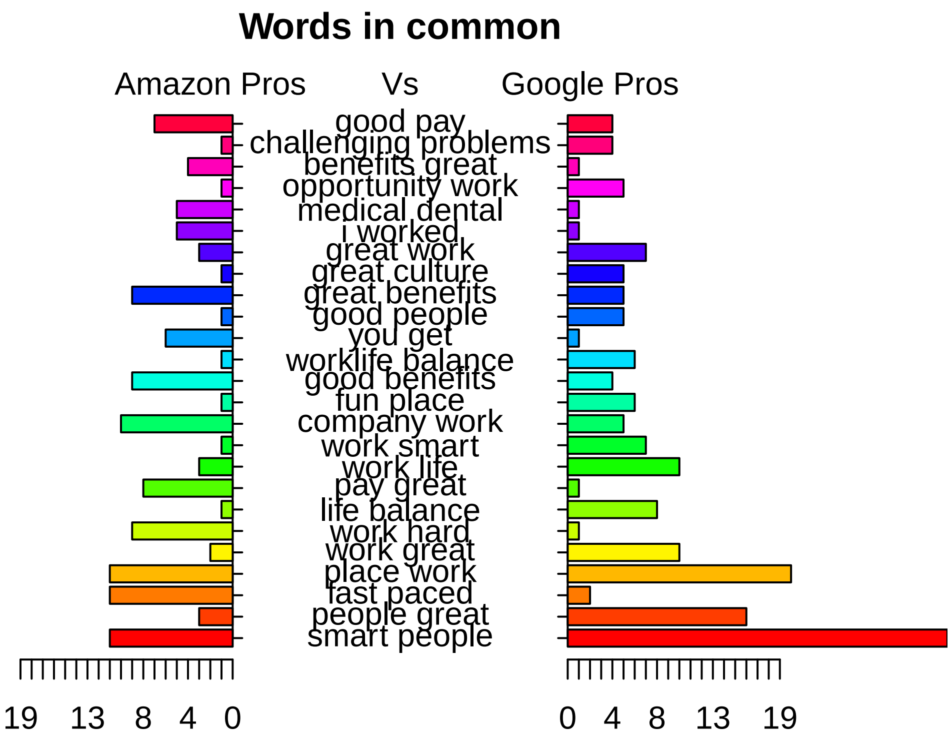
```
all_google_pros <- paste(google$pros,collapse="")
all_google_cons <- paste(google$cons,collapse = "")

all_google <- c(all_google_pros,all_google_cons)
all_google_qdap <- qdap_clean(all_google)
all_google_vs <- VectorSource(all_google_qdap)
all_google_vc <- VCorpus(all_google_vs)
all_google_clean<- tm_clean(all_google_vc)
all_google_tdm <- TermDocumentMatrix(all_google_clean)
colnames(all_google_tdm) <- c("Google Pros","Google Cons")
all_google_tdm_m <- as.matrix(all_google_tdm)

comparison.cloud(all_google_tdm_m,colors = c("orange","blue"),max.words = 50)
```

Amazon's positive reviews appear to mention bigrams such as "good benefits", while its negative reviews focus on bigrams such as "work-life balance" issues.

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

Now we'll make a pyramid plot lining up positive reviews for Amazon and Google so you can adequately see the differences between any shared bigrams.

```
amazon_pro <- paste(amazon$pros,collapse = "")
google_pro <- paste(google$pros,collapse = "")
all_pro <- c(amazon_pro,google_pro)
all_pro_qdap <- qdap_clean(all_pro)
all_pro_vs <- VectorSource(all_pro)
all_pro_vc <- VCorpus(all_pro_vs)
all_pro_corp <- tm_clean(all_pro_vc)

tdm.bigram = TermDocumentMatrix(all_pro_corp,control = list(tokenize =tokenizer))
colnames(tdm.bigram) <- c("Amazon","Google")
tdm.bigram <- as.matrix(tdm.bigram)
common_words<- subset(tdm.bigram,tdm.bigram[,1] > 0 & tdm.bigram[,2] > 0 )
difference <- abs(common_words[, 1] - common_words[,2])
common_words <- cbind(common_words,difference)
common_words <- common_words[order(common_words[,3],decreasing = TRUE),]
top25_df <- data.frame(x=common_words[1:25,1],y=common_words[1:25,2],labels=rownames(common_words[1:25,

library(plotrix)
pyramid.plot(top25_df$x,top25_df$y,labels=top25_df$labels,gap=15,top.labels=c("Amazon Pros","Vs","Googl
```

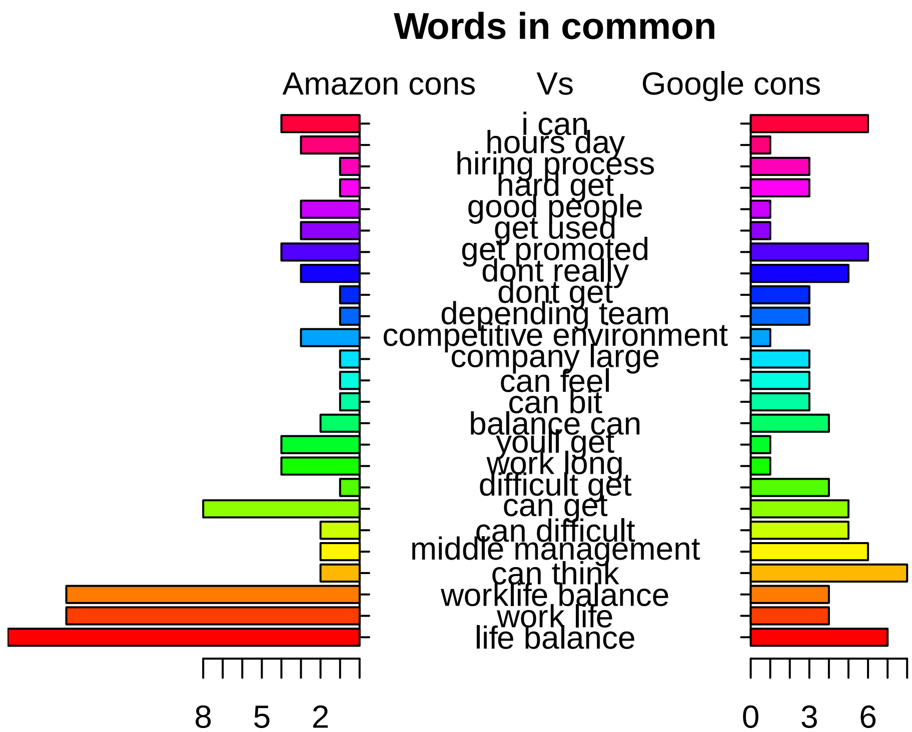



```
## [1] 5.1 4.1 4.1 2.1
```

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.

You now decide to turn your attention to negative reviews and make the same visuals.

```
amazon_cons <- paste(amazon$cons,collapse = "")
google_cons <- paste(google$cons,collapse = "")
all_cons <- c(amazon_cons,google_cons)
all_cons_qdap <- qdap_clean(all_cons)
all_cons_vs <- VectorSource(all_cons)
all_cons_vc <- VCorpus(all_cons_vs)
all_cons_corp <- tm_clean(all_cons_vc)

tdm.cons_bigram = TermDocumentMatrix(all_cons_corp,control=list(tokenize =tokenizer))

colnames(tdm.cons_bigram) <- c("Amazon","Google")
tdm.cons_bigram <- as.matrix(tdm.cons_bigram)
common_words<- subset(tdm.cons_bigram,tdm.cons_bigram[,1] > 0 & tdm.cons_bigram[,2] > 0 )
difference <- abs(common_words[, 1] - common_words[,2])
common_words <- cbind(common_words,difference)
common_words <- common_words[order(common_words[,3],decreasing = TRUE),]
top25_df <- data.frame(x=common_words[1:25,1],y=common_words[1:25,2],labels=rownames(common_words[1:25,]

library(plotrix)
pyramid.plot(top25_df$x,top25_df$y,labels=top25_df$labels,gap=10,top.labels=c("Amazon cons","Vs","Googl
```




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

We'll use Commonality cloud to show common between Aamazon and google with Unigram, Bigram and Trigram tokenizer to identify more insights.

Unigram

```
tdm.unigram <- TermDocumentMatrix(all_pro_corp)
colnames(tdm.unigram) <- c("Amazon","Google")
tdm.unigram <- as.matrix(tdm.unigram)

commonality.cloud(tdm.unigram,colors=c("red","yellow"),max.words = 100)
```

Bigram

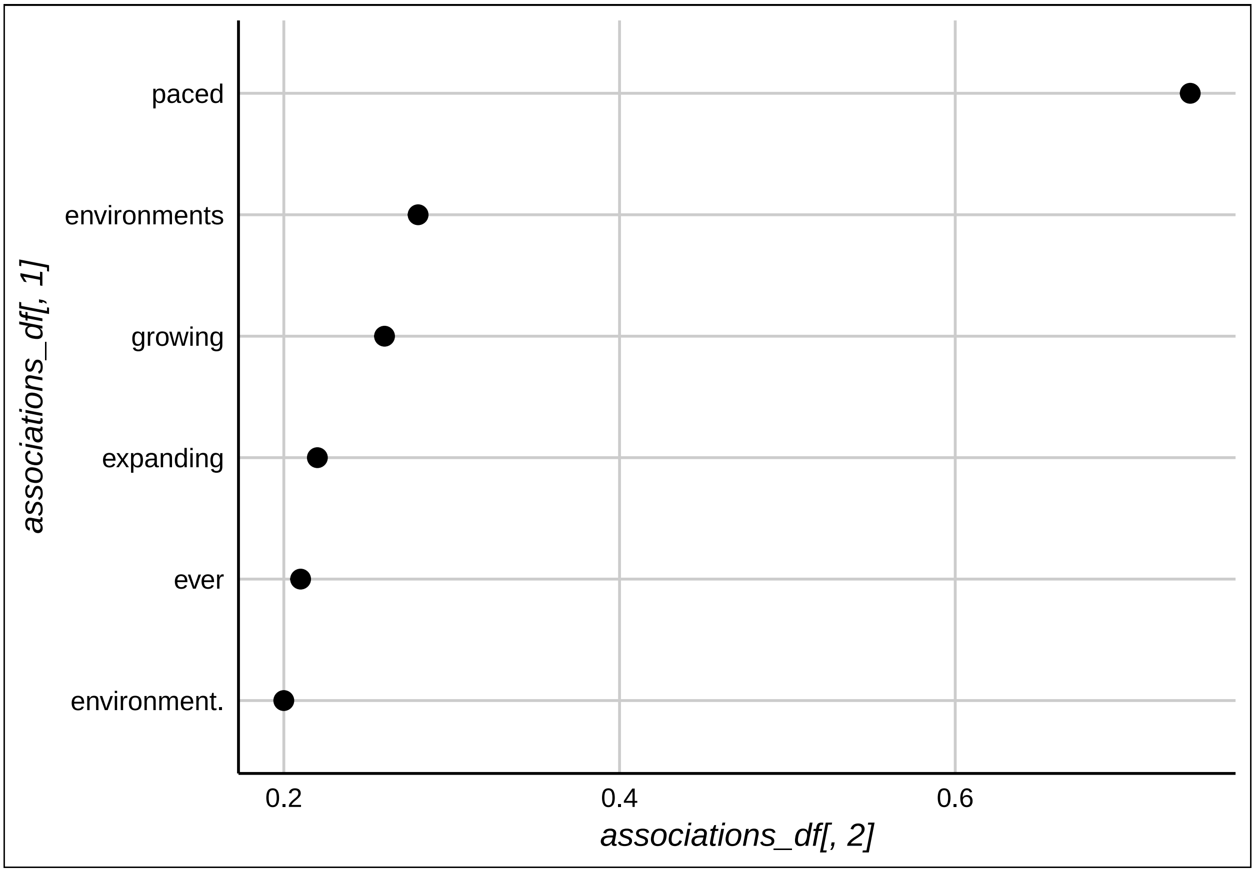
```
BigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
tdm.bigram <- TermDocumentMatrix(all_pro_corp,control = list(tokenize=BigramTokenizer))
colnames(tdm.bigram) <- c("Amazon","Google")
tdm.bigram <- as.matrix(tdm.bigram)

commonality.cloud(tdm.bigram,colors=c("red","yellow"),max.words = 100)
```

Trigram

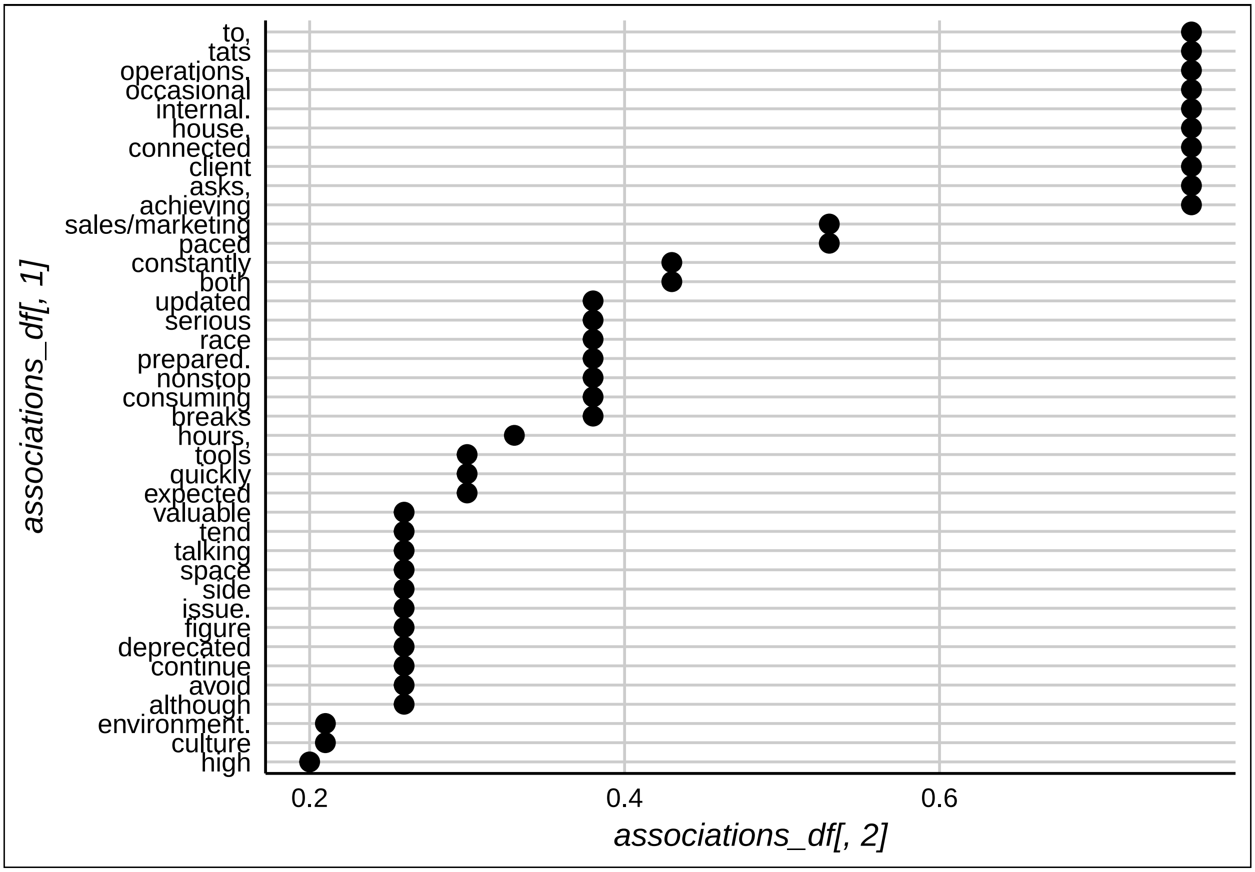
```r
TrigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))
tdm.trigram <- TermDocumentMatrix(all_pro_corp,control = list(tokenize=TrigramTokenizer))
colnames(tdm.trigram) <- c("Amazon","Google")
tdm.trigram <- as.matrix(tdm.trigram)

commonality.cloud(tdm.trigram,colors=c("red","yellow"),max.words = 100)
```

Plotting amazon and google association

```
library(ggthemes)
library(ggplot2)

amazon_tdm <- TermDocumentMatrix(amazon_p_corp)
associations <- findAssocs(amazon_tdm,"fast",0.2)
associations_df <- list_vect2df(associations)[,2:3]

ggplot(associations_df,aes(y=associations_df[,1]))+
  geom_point(aes(x=associations_df[,2]),
             data=associations_df,size=3)+
  theme_gdocs()
```




```
library(ggthemes)
library(ggplot2)

google_tdm <- TermDocumentMatrix(google_c_corp)
associations <- findAssocs(google_tdm,"fast",0.2)
associations_df <- list_vect2df(associations)[,2:3]

ggplot(associations_df,aes(y=associations_df[,1]))+
  geom_point(aes(x=associations_df[,2]),
             data=associations_df,size=3)+
  theme_gdocs()
```



Conclusion Google have a better work-life balance according to current employee reviews.

```
findAssocs(amazon_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
```

We Identified candidates that view an intense workload as an opportunity to learn fast and give them ample opportunity.