Ruchita Trivedi

Net Id: Zx-4954

Abstract

*Which company has better work life balance? Which has better perceived pay according to online reviews?*

Apple vs. google

Fight for acquiring top talent

Contents

[1 - Problem definition & specific goals 2](#_Toc473915839)

[1.1 Definition and Types: 2](#_Toc473915840)

[2 - Identify text to be collected 3](#_Toc473915841)

[3 - Text organization 4](#_Toc473915842)

[3.1 Cleaning 4](#_Toc473915843)

[Qdap package 4](#_Toc473915844)

[Tm package: 5](#_Toc473915845)

[Stemming and Stem Completion 6](#_Toc473915846)

[3.2 Preprocessing: 6](#_Toc473915847)

[Making Bigrams: 6](#_Toc473915848)

[4 - Feature extraction 7](#_Toc473915849)

[4.1 Word cloud: 7](#_Toc473915850)

[4.2 Comparison Cloud 9](#_Toc473915851)

[5. Analysis: 10](#_Toc473915852)

[5.1 Cluster Analysis using Dendogram: 10](#_Toc473915853)

[5.2 Finding Associations: 12](#_Toc473915854)

[5.3 Word Network: 13](#_Toc473915855)

[5.4 Pyramid Plot: 15](#_Toc473915856)

[6 - Reach an insight, recommendation or output 17](#_Toc473915857)

[7. Citations: 18](#_Toc473915858)

# 

# 1 - Problem definition & specific goals

Here is the case study , where I have pretended to work in HR Analytics and I have to help my team of recruiters with how external candidates perceive my organization compared to my competitor. I have pretended that I work for Apple and my biggest competitor being Google.   Google is the developer of the Android software, however, they are also a major competitor of Apple in the smartphone operating system market. In summary , the problem statement is :

*Which company has better work life balance? Which has better perceived pay according to online reviews?*

I have followed a basic map in transitioning from a unorganized state to an organized state, I order to reach a conclusion or a novel insight. In tech terms , Web Scrapping and Text Mining is applied.

## 1.1 Definition and Types:

**Web Scraping** (also termed Screen Scraping, Web Data Extraction, WebHarvesting etc.) is a technique employed to extract large amounts of data from websites whereby the data is extracted and saved to a local file in our computer or to a database in table (spreadsheet) format. The website used for scrapping is Indeed.com.

**Text mining,** also referred to as text data mining, roughly equivalent to text analytics, is the process of deriving high-quality information from text. High-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning.

**There are two approaches to text mining**

**Sematic Parsing**: In semantic parsing we care about word type and order. This method creates a lot of features to study . Example , a single word can be tagged part of a sentence ,then a noun and then a named entity. So one particular word has three features associated with it, which makes semantic parsing feature rich. To do the tagging semantic parsing follows a tree structure to continually break up the text.

**Bag of Words Method**: It doesn't care about word type or order. Here, words are just attributes of the document. Bag of words treat each word as a single token , no matter the size or the order.

I have used Bag of Words approach to text mining.

# 2 - Identify text to be collected

So, here I ve appled the text mining workflow to Apple and Google reviews gathered from Indeed.com from current and former employees of both companies. For gathering the positive and negative reviews of both the companies I have used the R's Rvest package.

Rvest is new package that makes it easy to scrape (or harvest) data from html web pages, inspired by libraries like [beautiful soup](http://www.crummy.com/software/BeautifulSoup/). It is designed to work with [magrittr](https://github.com/smbache/magrittr) so that we can express complex operations as elegant pipelines composed of simple, easily understood pieces. Install it with:

install.packages("rvest")

**rvest in action**

# We start by downloading and parsing the file :

library(rvest)

library(purrr)

url\_base <- "<https://www.indeed.com/cmp/Apple/reviews?fcountry=ALL>"

map\_df(1:100, function(i) {

# simple but effective progress indicator

cat(".")

pg <- read\_html(sprintf(url\_base, i))

data.frame(pros=html\_text(html\_nodes(pg, ".cmp-review-pro-text")),

cons=html\_text(html\_nodes(pg, ".cmp-review-con-text")),

stringsAsFactors=FALSE)

})->apple

write.csv(apple, "C:/Users/i037805/Desktop/apple.csv")

To extract the rating, we start with [selectorgadget](http://selectorgadget.com/) to figure out which css selector matches the data we want: .cmp-review-pro-text and .cmp-review-con-text. We use html\_node() to find the first node that matches that selector, extract its contents with html\_text(). We extracted pros and cons for a 100 web pages in the code above, then write the write the extracted pros and cons reviews to a.csv file . Imported the .csv file into R. We follow the same process for both Apple and Google reviews. Below, I ve shown the structure of loaded data for both companies:

> str(appl)  
'data.frame':        501 obs. of 3 variables:  
 $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
 $ pros: chr "Working at the Apple Culture. Great Experience" "Benefits, Stock plan, Discounts, Occasional free meals." "Fun, quick-paced, great pay" "benefits, peers" ...  
 $ cons: chr "last year working there, commuting to Elk Grove from Fremont" "Breaks are monitored closely, shifts are sporadically scheduled." "Slow advancement, customers tend to be a little off-put by prices" "work life balance" …

> str(appl\_pros)  
 chr [1:501] "Working at the Apple Culture. Great Experience" ...

> str(appl\_cons)  
 chr "last year working there, commuting to Elk Grove from Fremont Breaks are monitored closely, shifts are sporadically scheduled. S"| \_\_truncated\_\_

> str(goog)  
'data.frame':        501 obs. of 3 variables:  
 $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
 $ pros: chr "\* If you're a software engineer, you're among the kings of the hill at Google. It's an engineer-driven company without a doubt "| \_\_truncated\_\_ "1) Food, food, food. 15+ cafes on main campus (MTV) alone. Mini-kitchens, snacks, drinks, free breakfast/lunch/dinner, all day,"| \_\_truncated\_\_ "You can't find a more well-regarded company that actually deserves the hype it gets." "#NAME?" ...  
 $ cons: chr "\* It \*is\* becoming larger, and with it comes growing pains: bureaucracy, slow to respond to market threats, bloated teams, cros"| \_\_truncated\_\_ "1) Work/life balance. What balance? All those perks and benefits are an illusion. They keep you at work and they help you to be"| \_\_truncated\_\_ "I live in SF so the commute can take between 1.5 hours to 1.75 hours each way on the shuttle - sometimes 2 hours each way on a "| \_\_truncated\_\_ "- Google is a big company. So there are going to be winners and losers when it comes to career growth. Due to the high hiring b"| \_\_truncated\_\_ ...

> str(goog\_pros)  
 chr [1:501] "\* If you're a software engineer, you're among the kings of the hill at Google. It's an engineer-driven company without a doubt "| \_\_truncated\_\_ …

> 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, cros"| \_\_truncated\_\_ ...

# 3 - Text organization

## 3.1 Cleaning

Qdap package: Theqdap package offers other text cleaning functions. Each is useful in its own way and is particularly powerful when combined with the others.

* bracketX(): Remove all text within brackets (e.g. "It's (so) cool" becomes "It's cool")
* replace\_number(): Replace numbers with their word equivalents (e.g. "2" becomes "two")
* replace\_abbreviation(): Replace abbreviations with their full text equivalents (e.g. "Sr" becomes "Senior")
* replace\_contraction(): Convert contractions back to their base words (e.g. "shouldn't" becomes "should not")
* replace\_symbol() Replace common symbols with their word equivalents (e.g. "$" becomes "dollar")

library(qdap)

# Alter appl\_pros

appl\_pros<-qdap\_clean(appl\_pros)

# Alter appl\_cons

appl\_cons<-qdap\_clean(appl\_cons)

Loaded the text data as a vector called appl\_pros . Next step is to convert this vector containing the text data to a corpus. A 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 the 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, as a document. And the tm package provides what are called *Source* functions to do that. Next ,we pass it to another tm function, VCorpus(), to create our volatile corpus.

The VCorpus object is a nested list, or list of lists. At each index of the VCorpus object, there is a PlainTextDocument object, which is essentially a list that contains the actual text data (content), as well as some corresponding metadata (meta). It can help to visualize a VCorpus object to conceptualize the whole thing.

For example,  to examine the contents of the second review in ap\_p\_corp, we would subset twice. Once to specify the second PlainTextDocument corresponding to the second tweet and again to extract the first (or content) element of that PlainTextDocument

library(tm)

# Create ap\_p\_corp

ap\_p\_corp<-VCorpus(VectorSource(appl\_pros))

# Create ap\_c\_corp

ap\_c\_corp<-VCorpus(VectorSource(appl\_cons))

# Print the content of the 15th review in ap\_p\_corp

> ap\_p\_corp[[15]][1]  
$content  
[1] "Good pay and benefits"

Tm package: The tm package provides a special function tm\_map() to apply cleaning functions to a corpus. *Mapping* these functions to an entire corpus makes scaling the cleaning steps very easy.

* tolower(): Make all characters lowercase
* removePunctuation(): Remove all punctuation marks
* removeNumbers(): Remove numbers
* stripWhitespace(): Remove excess whitespace

removeWords() takes two arguments: the text object to which it's being applied and the list of words to remove.

Note that tolower() is part of base R, while the other three functions come from the tm package. Notice the use of content\_transformer to wrap the tolower() since its not from tm package.

tm\_clean <- function(corpus){

corpus <- tm\_map(corpus, removePunctuation)

corpus <- tm\_map(corpus, stripWhitespace)

corpus <- tm\_map(corpus, removeNumbers)

corpus <- tm\_map(corpus, content\_transformer(tolower))

corpus <- tm\_map(corpus, removeWords,

c(stopwords("en")))

return(corpus)

}

# Create appl\_pros\_corp

appl\_pros\_corp<-tm\_clean(ap\_p\_corp)

# Create appl\_cons\_corp

appl\_cons\_corp<-tm\_clean(ap\_c\_corp)

Stemming and Stem Completion**:** It's another important step in cleaning and preprocessing The StemDocument() uses a algorithm to segment words to their base.This helps aggregate terms, but often we are left with base tokens that are not words So, we need to take additional step to complete them to words. The StemCompletion() takes as arguments the stemmed words(or tokens) and a dictionary of complete words. Here, we dropped the idea of using Stemming and Stem completion because we used bigrams for analysis.

## 3.2 Preprocessing:

Making Bigrams: The default is to make TDMs with unigrams, but here we have focused on tokens containing two or more words. This can help extract useful phrases which lead to some additional insights or provide improved predictive attributes for a machine learning algorithm.

The function below uses the RWeka package to create bigram (two word) tokens: min and max are both set to 2.

tokenizer <- function(x)

NGramTokenizer(x, Weka\_control(min = 2, max = 2))

Then the customized tokenizer() function can be passed into the TermDocumentMatrix or DocumentTermMatrix functions as an additional parameter. A Term Document Matrix has each word as rows and document as colomns.

appl\_p\_tdm <-TermDocumentMatrix(appl\_pros\_corp, control = list(tokenize = tokenizer))

> appl\_p\_tdm  
<<TermDocumentMatrix (terms: 142, documents: 501)>>  
Non-/sparse entries: 1935/69207  
Sparsity : 97%  
Maximal term length: 25  
Weighting : term frequency (tf)

# 4 - Feature extraction

## 4.1 Word cloud:

We do feature extraction by making visuals. There are many reasons that visualization is an important skill to a text miner. Chief amongst them is that

* good visualizations help decision makers come to quick conclusions. This is because
* human brain efficiently processes visual information.

In order to analyze a TDM we need to change it to a simple matrix using as.matrix.

Calling rowSums() on our newly made matrix aggregates all the terms used in a passage. Once we have the rowSums(), we can sort() them with decreasing = TRUE, so we can focus on the most common terms.

appl\_p\_tdm\_m<-as.matrix(appl\_p\_tdm)

appl\_p\_freq<-rowSums(appl\_p\_tdm\_m)

appl\_p\_freq1<-as.matrix(appl\_p\_freq)

wordcloud(names(appl\_p\_freq), appl\_p\_freq, max.words= 50,scale=c(2,.2),rot.per=.5, colors=pal2)

variov medical 
disabilty insurance 
wonderfully designed 
discounts apple E 
environment convenient o -o 
convenient hours 
good pay O 
hours relaxed 
insurance dental o 'D paid holidays Q) 
3 CO dental insurance u E 
vision insurance 
Q) high entry 
benefits stock 
o Edesignea environment Q 
worklite balance c 
great benefits 
great people 
access various 
3 
people wonderfully 
fun quickpacad 
o 

*Looks like most people reported Apple to have a great culture , great people , great pay.*

Similarly , we created a word cloud to look into the most reported negative bigrams.

wordcloud(names(appl\_c\_freq) ,appl\_c\_freq, max.words=30,scale=c(2,.2),rot.per=.5,color=brewer.pal(10, "PuOr"))

working commuting 
commuting elk 
o 
g except thanksgiving-a 
werk every per position a 
ß) CD 
(D 
worked hr 
every holiday 
grove fremont 
possessive cullure{d 
(D 
scheduleretail life 
year working 

*Looks like most people are unhappy about commuting to "elk Groove" from "Fremont". They want to work remotely My team team might want to suggest allowing remote work for tech staff. Also people find "advancement difficult" and "salary lower". They also complain of not getting enough holidays "except thanksgiving". They also seem unhappy about "position politics" ,"relentless pushing"………phew!!*

## 4.2 Comparison Cloud

We decided to create a comparison.cloud() of Google's positive and negative reviews for comparison to Apple. This gave us a quick understanding of top terms without having to spend as much time as i did examining the Apple reviews .

# Build a comparison cloud

comparison.cloud(all\_m, colors=c("#F44336", "#2196f3") ,max.words = 100,scale=c(2,.2))

ern vi r-r-l e n t 
ben efi ts g 00 d 
fu 
i tic s 
g re a t 
g et 
ha rd time, 

*Seems like good food, perks, good environment , amazing people work culture were the top reported pros of Google . While beurocracy, politics,  "getting big", "bureaucracy", and "middle management", "competition " were among the most reported cons of google. Note that, there were lesser negatives (blues)compared to positives (reds)reported for google. Many even reported "nothing" as negative , working in google.*

# 5. Analysis:

## 5.1 Cluster Analysis using Dendogram:

It seems there is a strong indication of "long commuting" and "poor pay" in my company's reviews. As a simple clustering technique, we decide to perform a hierarchical cluster and create a dendrogram to see how connected these phrases are.

A simple way to do word cluster analysis is with a dendrogram on our term-document matrix. Once we have a TDM, we can call dist() to compute the differences between each row of the matrix.

Next, we call hclust() to perform cluster analysis on the dissimilarities of the distance matrix. Lastly, we can visualize the word frequency distances using a dendrogram and plot(). Often in text mining, we can tease out some interesting insights or word clusters based on a dendrogram.

 But first, we have to limit the number of words in our TDM using removeSparseTerms() from tm. TDMs and DTMs are sparse, meaning they contain mostly zeros. We won't be able to easily interpret a dendrogram that is so cluttered, especially if we are working on more text.

A good TDM has between 25 and 70 terms. The lower the sparse value, the more terms are kept. The closer it is to 1, the fewer are kept.( This value is a percentage cutoff of zeros for each term in the TDM.)

library(dendextend)

# Create appl\_c\_tdm2 by removing sparse terms

appl\_c\_tdm2<-removeSparseTerms(appl\_c\_tdm,sparse=.96)

# Create hcd as a dendogram having cluster of distance values

hcd<-as.dendrogram(hclust(dist(appl\_c\_tdm2, method = "euclidean"),method="complete"))

labels(hcd)

hcd<- branches\_attr\_by\_labels(hcd, c("commuting elk", "salary lower"),color="red")

# Produce a plot of hc

plot(hcd, main= "Dendogram showing negative bigram clusters" )

rect.dendrogram(hcd, k=3, border="blue")

# 

2 
4 
6 
earwor In 
commu In 
ast year 
grove fremont 
:ommuting elk 
sa arv owe 
advantemen 
per position 
eme t difficult 
u ere al 
worked hr 
werk every 
hr days 
101iday except 
t thanksgiving 
da swerK 
wc? %frlaidaaæ 
30mpensatipn 
dule flexibility 
ture schefiule 
low level 
advance low 
ficult advance 
aro nd schoo 
8 
o 
2. 

"Salary Lower" and "Commuting elk " were in different clusters in dendogram.

*Dendogram clearly shows that some people experience problems related to commute to "Elk Groove". These people mostly live at "Groove Fremont" . These people have a bad taste because its been a "year working" and they are "working comuting" . My HR team can suggest their project managers to allow working remotely.*

*Another cluster in dendogram shows that some people are experiencing problems related to "low salary", and find "advancement (promotion) difficult". They feel their salary could be higher. My HR team could consult them to understand their expectations.*

As expected, we only see similar topics throughout the dendrogram.

## 5.2 Finding Associations:

*As we have seen that there as a lot of negative sentiment regarding salary, whereas we have also seen that "great salary" was reported by most people as positive review. Lets investigate this aspect further and understand this contradiction. Below, we have tried to find a interesting correlation between the said ,*

Another good way to think about word relationships is with the findAssocs()function in the tm package. For any given word, findAssocs()calculates its correlation with every other word in a TDM or DTM. Scores range from 0 to 1. A score of 1 means that two words always appear together, while a score of 0 means that they never appear together. We have applied findAssoc() to positive reviews ,

# Find associations with great pay

associations<-findAssocs(appl\_p\_tdm ,"great pay", 0.2)

# View the venti associations

associations

# Create associations\_df

associations\_df<- list\_vect2df(associations)[, 2:3]

require(ggplot2)

install.packages("ggthemes")

library(ggthemes)

# Plot the associations\_df values (don't change this)

ggplot(associations\_df, aes(y = associations\_df[, 1])) +

geom\_point(aes(x = associations\_df[, 2]),

data = associations\_df, size = 3) +

theme\_gdocs()

assocjabons 0 〃 
a 0 ロ a00 21 

*The ggplot above provides answer to our confusion above .A "great salary" has a high correlation with "benefits incentives", "bonuses great", "healthcare benefits", "incentives bonuses", "payhome". Ie People who are satisfied with their salary were ones who receive additional perks like bonuses, healthcare benefits , paid home , incentives etc. Quite understandable!*

## 5.3 Word Network:

Another way to view word connections is to treat them as a network, similar to a social network. Word networks show term association and cohesion. In a network graph, the circles are called *nodes* and represent individual terms, while the lines connecting the circles are called *edges* and represent the connections between the terms.  qdap provides a method for making word networks. The word\_network\_plot()and word\_associate() functions both make word networks easy.

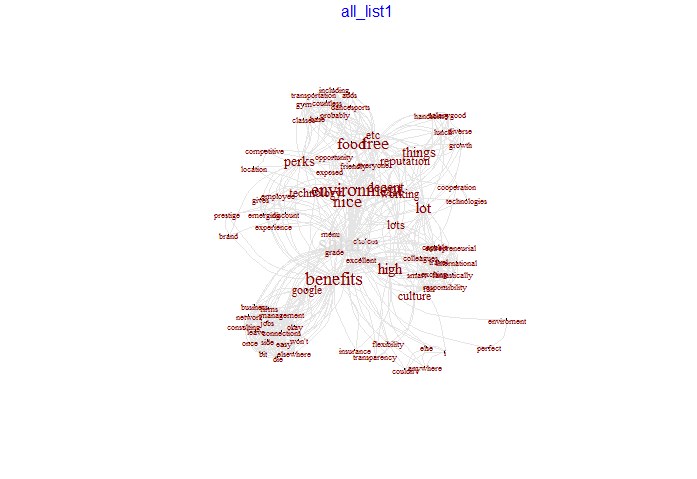
*We decided to try our hands at this but this visual became very dense and hard to interpret visually. Thus, not very insightful for our analysis.*

# Word association

word\_associate(goog\_pros, match.string = c("salary"),

stopwords = c(Top200Words, "company", "great"),

network.plot = TRUE, cloud.colors = c("gray85", "darkred"))



## 

## 5.4 Pyramid Plot:

Apple's positive reviews appear to mention bigrams such as "great culture", while its negative reviews focus on bigrams such as "salary low" and " long commute " issues.

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

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

 We created a simple matrix from the Apple and Google positive reviews using all\_tdm\_m<-as.matrix(pros\_tdm). This matrix contains three columns: terms, term frequency in the google\_pros corpus, term frequency in the appl\_pros corpus. So we can use the subset() function in the following way to get terms that appear one or more times in both corpora:

# Create common\_words

common\_words <- subset(all\_tdm\_m, all\_tdm\_m[, 1] > 0 & all\_tdm\_m[, 2] > 0)

Once we have the terms that are common to both corpora, we can create a new column in same\_words that contains the absolute difference between how often each term is used in each corpus.

To identify the words that differ the most between documents, we must order() the rows of same\_words by the absolute difference column with decreasing = TRUE like this:

common\_words <- common\_words[order(common\_words[, 3], decreasing = TRUE), ]

# Create difference

difference <- abs(common\_words[, 1] - common\_words[, 2])

# Add difference to common\_words

common\_words <- cbind(common\_words, difference)

# Order the data frame from most differences to least

common\_words <- common\_words[order(common\_words[, 3], decreasing = TRUE), ]

common\_words

Now that same\_words is ordered by the absolute difference, we create a small data.frame() of the 10 top terms so we can pass that along to pyramid.plot():

# Create top15\_df

top15\_df <- data.frame(x = common\_words[1:15, 1],

y = common\_words[1:15, 2],

labels = rownames(common\_words[1:15, ]))

Note that top15\_df contains columns x and y for the frequency of the top words for each of the documents, and a third column, labels, that contains the words themselves.

Finally, we created the pyramid.plot() to get a better feel for how the word usages differ by company.

Google 
Words in Common 
Pro Words 
food gym 
benefits great 
good pay 
health benefits 
free meals 
excellent benefits 
health insurance 
work good 
ood communicatio 
nice perks 
free food 
great people 
great pay 
great experience 
Apple 
41 
35 
29 
23 
17 
11 
6 
2 
culture great 
0 
4 
8 
13 
19 
25 
31 
37 

*I was happy and surprised to see that the common pros were much more in my company Apple than Google. Whereas, I had the perception that Google is more attractive for employees than Apple . Except for more "free food " at Google . My company was much better at everything else including Salary . Probably, I will suggest my team to increase budget on food items.*

Similarly we follow the same steps for Negative reviews for more insights.

Words in Common 
Google Cons Words Apple 
worklife balance 
really flexible 
politics promotions 
development none 
work life 
life balance 
poor management 
job security 
lary compensatio 
difficult advance 
15 
5 
0 
4 
8 
13 

This pyramid plot surprised me again. While the "worklife balance" complains were about the same in both companies." Work life" was more stressed for Apple employees comparatively and they were not able to achieve "life balance". Also there were more issues related to "promotion politics" and people really thought that the "development none"(development was none). In other words, It was a "difficult advance" or difficult promotions etc at Apple. Issues related to "poor management "and "Job Security" were also more. And , as we expected low "Salary Compensation " was most reported.

*I think people are stressed about job-security, that takes away their work-life balance. I will suggest my team to take measures to improve work-life balance at my company .*

*As a bottom-line, I can tell that people at Apple enjoy more benefits but at the cost of dis-balanced work-life.*

# 6 - Reach an insight, recommendation or output

*Wordclouds for Apple's positive reviews in appear to mention bigrams such as "great culture", "great people", while its negative reviews focus on bigrams such as "low salary" and "commuting elk" issues. Later Word Association plot showed that low salary was compensated with incentives, paid home, bonuses etc. for deserving employees. Will ask my HR Team to suggest to management regarding allowing remote work, wherever possible.*

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

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

*In Pros Words Bigram, I was happy and surprised to see that the common pros were much more in my company Apple than Google. Whereas, I had the perception that Google is more attractive for employees than Apple . Except for more "free food " at Google . My company was much better at everything else including Salary . Probably, I will suggest my team to increase budget on food items.*

*In Cons Words Bigram, While the "worklife balance" complains were about the same in both companies." Work life" was more stressed for Apple employees comparatively and they were not able to achieve "life balance". Salary and Work-Life Balance are two most important factors that contribute to employee satisfaction . As a HR officer, I need to report about these imbalanced work life issues in my company. there were more issues related to "promotion politics" and people really thought that the "development none"(development was none). In other words, It was a "difficult advance" or difficult promotions etc at Apple. Issues related to "poor management "and "Job Security" were also more. People really look tensed because of such work conditions, my team needs to work out ways to make the work environment healthy . And , as we expected low "Salary Compensation " was most reported but we know that management is compensating the deserving ones in other ways.*

*In nutshell I would conclude that, Apple employees were getting paid more but Google employees had a much better work-life balance.*

# 7. Citations:

1. <http://stackoverflow.com/questions/40597572/web-scraping-across-multiple-pages-with-rvest>
2. <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0ahUKEwiEqbuyx_XRAhUgHGMKHV-JCkQQFggzMAQ&url=https%3A%2F%2Feight2late.wordpress.com%2F2015%2F05%2F27%2Fa-gentle-introduction-to-text-mining-using-r%2F&usg=AFQjCNF0pZ02pehnyQWtNoE1Am8-AqW0VQ&sig2=gkDJBhxerbUQA-ESs4njGA>
3. <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=8&cad=rja&uact=8&ved=0ahUKEwiEqbuyx_XRAhUgHGMKHV-JCkQQFghHMAc&url=http%3A%2F%2Fworldcomp-proceedings.com%2Fproc%2Fp2015%2FDMI8036.pdf&usg=AFQjCNHeUwEmhcBFF3_qQFu3tCFJWGl6kA&sig2=znaLYRUkm5e4rRn4J0kD9Q>
4. <http://www.kdnuggets.com/2017/01/data-mining-amazon-mobile-phone-reviews-interesting-insights.html>