CASE STUDY REPORT
Data Mining in Engineering
Saurabh Yelne

Executive Summary:

Our project pertains to HR analytics with our main objective to uncover a pattern in Amazon's negative reviews. Amazon has continually been one of the leaders in high attrition. This comes as a surprise given its reputation as one of the most innovative and successful companies. The results of this study can be used by Amazon to adapt their business model in order to accommodate their employees and reduce attrition.

The data leveraged for this study were glass door reviews labeled as Pros and Cons. The data has been preprocessed to remove URLs, symbols, stop words, and numbers. The data pattern was initially uncovered by using a simple word frequency chart, indicating words such as "hours", "life", "training", and "managers" appear more in negative reviews. In order to prove the pattern is significant, we developed three models. We trained naïve Bayes as our base model and then used SVM and bagging as the other two models for predicting the correct sentiment. We achieved 27% accuracy with naive Bayes, 73% accuracy with SVM, and 74% bagging.

Only one model exceeded the naïve rule (73%), which is the bagging model. This suggests negative and positive reviews do not differentiate significantly with respect to word usage. Unfortunately, this entails the words found in the negative reviews cannot be pushed on Amazon and we cannot definitively suggest them to make any changes. With a larger dataset, it is possible to redo the study; however, currently, the results are inconclusive.

I. Background and Introduction

Employee turnover is a natural part of business in any industry. However excessive turnover decreases the overall efficiency of the company and comes with a serious price tag. Understanding the effects of losing a high number of employees serves as a motivator towards improving profits and creating a more appealing work environment. Each employee who resigns costs the company money. Studies have shown that companies pay up to 30% of yearly salary in new employee costs. Investment in employees becomes expensive through advertising the vacancy, moving expenses, training, benefits and so on.

Amazon, one of the largest retail companies as of 2017 only after CVS and Walmart., is a prime example of a company the suffers from attrition. The unconventional retailer, like other tech companies, prides itself on its intelligent employees; however, even with the high pay and benefits, Amazon continually struggles with employee retention. In fact, the company came in second in a 2013 list for the shortest employee tenure.

Rank	Employer Name	Median Age of Employees	Median Employee Tenure	Median Pay
1	Massachusetts Mutual Life Insurance Company	38	0.8	$60,000
2 - tie	Amazon.com Inc	32	1.0	$93,200
2 - tie	American Family Life Assurance Company of Columbus (AFLAC)	38	1.0	$38,000
4 - tie	Google, Inc.	29	1.1	$107,000
4 - tie	Mosaic	37	1.1	$69,900

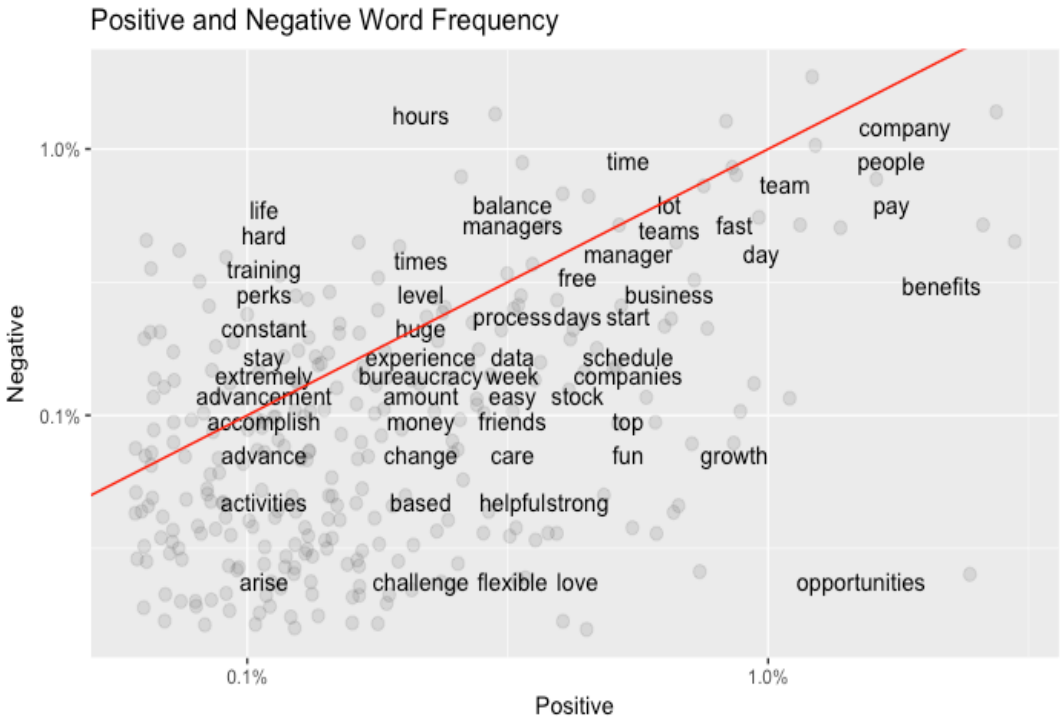
To understand why the giant retailer is struggling, we need to understand the sentiments of workers currently or previously at Amazon. The data set for Amazon that we leveraged can be found on Glassdoor. For the purpose of this study, we have aggregated 500 reviews from glass door and have labeled them either pro or con. This is pivotal, given that all classification methods used for this study will need to be labeled in order to execute sentimental analysis.

The overall goal of the study is to observe the differences in words between positive and negative reviews and determine if the difference is significant. If certain words are repeated enough within the negative reviews, in theory, our models should capture that and correctly classify the reviews as negative. However, if there is no relationship in terms of words and positive or negative reviews, our models will not work.

II. Data Exploration and Visualization

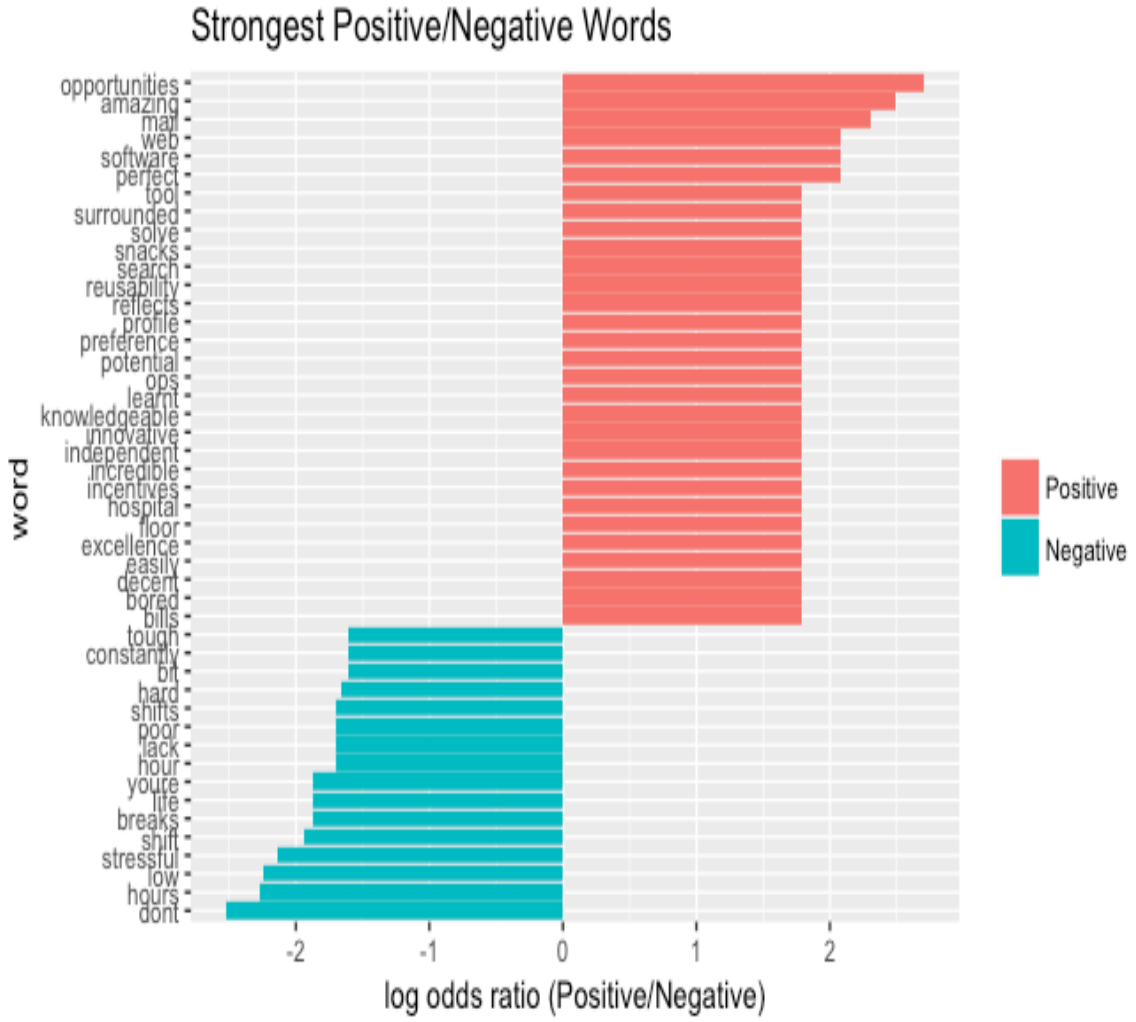
From a data collection standpoint, we noticed trends immediately in the positive and negative reviews. It appeared previous employees were not fond of the hours and/or management. We wanted to see if we could capture this trend through visual methods.

Below we produced a graph depicting the negative word frequency vs the positive word frequency. The graph was constructed by counting all of the distinct words for all reviews. In total, there were 3,607 distinct words (excluding stop words) within the negative reviews and only 928 within the positive reviews. We then divided how often each word appeared by the total distinct words, producing the frequency percentage. As we can see, hours are clearly more prevalent in the negative reviews than it is in the positive reviews. Management is difficult to tell, as "managers" is more negative than positive but "manager" is more positive than negative.



To get further insight, we created a chart depicting the log odds ratio for each word. From here we are able to see which words have the highest probability of being classed as negative or positive. The positive words coincide with our expectations, as Amazon is known for its innovative, knowledgeable software teams with ample opportunities. The negative

reviews depict a different picture, as it illustrates a stressful work environment with long hours and little breaks.



III. Data Preparation and Preprocessing

Prior to exploring the data, we needed to clean it. Data preprocessing is essential in text mining. Cleaning in the text mining refers to removing all words, symbols, or numbers that can interfere with modeling the dataset. The reason for this is because the model should only be fed words that carry sentiment. With regards to data exploration and data modeling, we had two separate approaches for cleaning the data.

For data exploration, our end goal was to have a matrix with individual words grouped by their sentiment, including their frequency. It is important to note that this dataset does not keep the actual reviews, as it was intended to capture an overall picture of the sentiment and relationship with certain words. First, we “unnest” the dataset, which

entails changing the review column into a words column. Every review is deconstructed into multiple rows of words. Within the "unnest" step, we also remove any symbols and numbers from those reviews. Next, we remove all stop words. Stop words are words such as "the", "of", or "a" that indicate no feeling or carry no sentimental value. We also removed words that we expect to appear often in both positive and negative reviews. In this case, the only words filtered out are ones that begin with "amazon". Finally, we aggregate the words at sentiment level, calculating their frequency.

For modeling, we needed a term document matrix, in order to keep track of the reviews and their sentiment. Similarly, to the previous dataset, we remove the symbols and punctuation. The removal of symbols and numbers is done in R by calling the base function "gsub()". The gsub function takes a character vector for a given pattern and replaces the characters. In our case, we will be replacing the characters with nothing; thus, removing all the unnecessary symbols/punctuation and numbers. Next, we partition the dataset. To partition the data set, we will filter for 60% of the positive and 60% of the negative reviews and assign it to the training set. The remainder of the reviews will be leveraged for testing our classification model. We then combined the training and testing data back together; however, now they have been ordered accordingly. We then transformed the dataset into a term document matrix (TDM). The term document matrix is a matrix where the reviews are the row names, the words from the reviews become the column names, and their frequency becomes their values. Given that we do not care about words that only appear once, we have set the minimum frequency to two so words not appearing at least twice will be removed from the column names. Within this step, all stop words are removed as well.

IV. Data Mining Techniques and Implementation

Given our term document matrix, we believe the best models to fit our data are Naïve Bayes, Support Vector Machine (SVM), and Bagging model. These models closely align to text classification and can easily be implemented using the "RTextTools" package.

Naïve Bayes

Naive Bayes classifiers are probabilistic classifiers based on Bayes theorem. It assumes that the features in a dataset are mutually independent and Naïve Bayes performs

very well under this assumption. For smaller datasets, naive Bayes classifiers can perform better than other powerful options.

The text categorization with Naïve Bayes works well due to the assumption that features are independent at the word level. When the number of attributes are large, the independence assumption allows for the parameters of each attribute to be learned separately, simplifying the learning process.

Naïve Bayes has two different event models. The multi- variate model uses a document event model, with the binary occurrence of words being attributes of the event. However, the model doesn't take into account the multiple occurrences of words within the same document. If multiple word occurrences are to be taken into consideration, then a multinomial model should be used, where a multinomial distribution accounts for multiple word occurrences. Here, the words are the events.

Support Vector Machine

For text mining operations like topic assignment and filtering, SVMs are used extensively in many and so SVM applications that are structured for sparse data matrices is important. The sparse matrix formats are supported by support vector machine and also solve the dual optimization problem by utilizing algorithms like SMO (Platt (1999)) in R.

SVM is a new technique for multi-dimensional function approximation. The objective of support vector machines is to determine a classifier or regression function, which minimizes the based on the risk of training set error and the confidence interval which corresponds to the generalization or test set error.

Bagging

Bagging algorithm is one of the most common integrated learning algorithms. The basic idea is that a training set and a weak learning method is given, several rounds of training in the training set points (set T rounds), each round of training set is composed by n randomly selected training samples from the original training sets. A prediction function hi is obtained after each round of training. At last T prediction functions is obtained: h1, h2,..., hT. Using the predicted sequence of functions gives a prediction on the sample set, and then get the final prediction h* by majority voting rules. The choice of weak classifier in Bagging algorithm is required. Bagging can effectively improve the classification accuracy for an

instable weak learning algorithm, however, the stable learning algorithm has little effect on classification accuracy, even reducing it sometimes.

After the text is represented into another kind of format, which can be analyzed mathematically, the next task is to use the appropriate algorithm for training text on the basis of this mathematical model.

V. Performance Evaluation

Naïve Bayes resulted in an underperforming model. The accuracy stands at .27, which suggests that in terms of prediction, the naïve rule is better suited for this data. The naïve rule would predict all reviews as negative and achieve 0.73 accuracy.

Model:	Naïve Bayes		
Accuracy:	0.27		
		Negative	**Positive**
	Negative	0	100
	Positive	0	37

The output from the SVM model is the naïve rule such that all reviews are predicted as negative. Given that a model should exceed the naïve rule, this model is also considered underwhelming even though the accuracy is high.

Model:	SVM		
Accuracy:	0.73		
		Negative	**Positive**
	Negative	100	0
	Positive	37	0

Finally, the bagging model performs slightly better than the previous two achieving a 0.74 accuracy. Although better than the previous two, it is still not apparent if it is significantly different than the naïve rule.

Model:	Bagging		
Accuracy:	0.74		
		Negative	**Positive**
	Negative	99	1
	Positive	35	2

VI. Discussion and Recommendation

Based on the bagging model achieving slightly above the naïve rule, we were able to capture some relationship between the words and the reviews. Although the models seemed appropriate, a lack of data might have been at fault regarding fitting the model, as we were unable to successfully classify the reviews. Since we cannot prove there is a significant difference between the reviews, we cannot assert that the attrition is made up of one or two factors. We must assume that attrition is made up of multiple reasons, which is why the model cannot classify the reviews correctly.Besides data, our model was limited with respect to n grams. N grams refers to the successive words captured in the data. For our model, we only considered individual words, which might have strayed the model away from bigrams such as “work life” or “fast paced”. Below we did a quick word cloud for the negative reviews using bigrams:

Unsurprisingly, the above cloud depicts similar results found in the data exploration section. If the model were able to classify the data successfully, Amazon would able to leverage the word frequencies and respond accordingly, ensuring their employees are working less hours in order to maintain a proper work life balance.

VII. Summary

The initial goal of the study was to identify key words within negative reviews and justify they are significant (constantly repeating). By identifying these words, we could understand why Amazon has such high attrition. However, to prove significance, we decided to use three models, which would need to successfully classify the reviews.

Prior to inputting the reviews into the model, we cleaned and explored the data. We removed all URLs, symbols, and stop words. Next, we analyzed the distribution of the words between the negative and positive reviews. From an initial standpoint, there did seem to be a difference in word usage between the two reviews. As stated earlier, “hours” was more prevalent in the negative reviews. This lead to the assumption that the models would capture this and label a review negative most likely if “hours” appeared in it.

By training the model on the training set, we expected three models that would achieve better accuracy than the naïve rule. Unfortunately, only one model succeeded in having better accuracy than the naïve rule. The bagging model that the difference may be significant; however more data is necessary to carry the study even further.

Overall, the results are inconclusive. Neither model successfully disproved or proved that the difference in word usage amongst the positive and negative reviews were significant. This could also entail Amazon does not have one major underlying issue. It could be a plethora of minor/personal issues or it could simply be that the hires from Amazon are in demand throughout the retail market.

Appendix: R Code for this case study

```{r library}
library(tidyverse)
library(tidytext)
library(stringr)
library(scales)
library(RTextTools)
library(e1071)
library(readxl)
```
```{r Data}
#Original Dataset
amazon_sentiment <- read_xlsx("/Users/JosephVele/Documents/R
Code/Data_mining/Amazon.xlsx",
                col_names = TRUE)
```

#DataPreprocessing
```{r DAta Pre}
amazon_sentiment$text <- gsub("[^[:alpha:][:space:]]*", "", amazon_sentiment$text)
amazon_sentiment$text <- gsub("http[^[:space:]]*", "", amazon_sentiment$text)
 #Set train and test data
 train_data <- rbind(filter(amazon_sentiment, Sentiment=='Negative')[1:300,],
                filter(amazon_sentiment, Sentiment=='Positive')[1:112,])
 test_data <- rbind(filter(amazon_sentiment, Sentiment=='Negative')[301:400,],
                filter(amazon_sentiment, Sentiment=='Positive')[113:149,])
 #because of previous step we can rbind them and know they have been separated
 total_data <- rbind(train_data, test_data)
 #Build Term Document Matrix
 matrix<- create_matrix(total_data[,1], language="english",
                  minDocFreq = 2,removePunctuation = TRUE,
                  removeStopwords=TRUE, removeNumbers=TRUE,
                 stemWords=TRUE )
```

#Data Visualization and Exploration
```{r visualization}
replace_reg <- "https://t.co/[A-Za-z\\d]+|http://[A-Za-z\\d]+|&amp;|<|>|RT|https"
unnest_reg <- "([^A-Za-z_\\d#@']|'(?![A-Za-z_\\d#@]))"
tidy_data <- amazon_sentiment %>%
 filter(!str_detect(text, "^RT")) %>%
 mutate(text = str_replace_all(text, replace_reg, "")) %>%
 unnest_tokens(word, text, token = "regex", pattern = unnest_reg) %>%
 filter(!word %in% stop_words$word,
      str_detect(word, "[a-z]"))%>%
 filter(substr(word,1,6)!="amazon")
```

```
frequency <- tidy_data %>%
  group_by(Sentiment) %>%
  count(word, sort = TRUE) %>%
  left_join(tidy_data %>%
            group_by(Sentiment) %>%
            summarise(total = n())) %>%
  mutate(freq = n/total)


frequency <- frequency %>%
  select(Sentiment, word, freq) %>%
  spread(Sentiment, freq) %>%
  arrange(Positive, Negative)

ggplot(frequency, aes(Positive, Negative)) +
  geom_jitter(alpha = 0.1, size = 2.5, width = 0.25, height = 0.25) +
  geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
  scale_x_log10(labels = percent_format()) +
  scale_y_log10(labels = percent_format()) +
  geom_abline(color = "red")+
  ggtitle("Positive and Negative Word Frequency")

word_ratios <- tidy_data  %>%
  count(word, Sentiment) %>%
  filter(sum(n) >= 10) %>%
  ungroup() %>%
  spread(Sentiment, n, fill = 0) %>%
  mutate_if(is.numeric, funs((. + 1) / sum(. + 1))) %>%
  mutate(logratio = log(Positive / Negative)) %>%
  arrange(desc(logratio))

word_ratios %>%
  arrange(abs(logratio))

word_ratios %>%
  group_by(logratio < 0) %>%
  top_n(15, abs(logratio)) %>%
  ungroup() %>%
  mutate(word = reorder(word, logratio)) %>%
  ggplot(aes(word, logratio, fill = logratio < 0)) +
  geom_col() +
  coord_flip() +
  ylab("log odds ratio (Positive/Negative)") +
  scale_fill_discrete(name = "", labels = c("Positive", "Negative"))
```

#DataMiningTechniques
```{r Data Mining}

# train the model
mat <- as.matrix(matrix)
classifier <- naiveBayes(mat[1:412,], as.factor(total_data$Sentiment[1:412]) )

# test the validity
predicted <- predict(classifier, mat[413:549,])

table(total_data$Sentiment[413:549], predicted)
recall_accuracy(total_data$Sentiment[413:549], predicted)

# build the data to specify response variable, training set, testing set.
container <- create_container(matrix, as.numeric(as.factor(total_data$Sentiment)),
                     trainSize=1:412, testSize=413:549,virgin=FALSE)

models <- train_models(container, algorithms=c("SVM", "BAGGING")
results <- classify_models(container, models)

# accuracy table
table(total_data$Sentiment[413:549], results[,"BAGGING_LABEL"])
table(total_data$Sentiment[413:549], results[,"SVM_LABEL"])
# recall accuracy

#wordcloud
library(readxl)
library(tm)
library(qdap)
library(sentimentr)
library(tidytext)
library(NLP)
library(wordcloud)
library(data.table)

ama_con <- read_excel("~/Amazon.xlsx")
str(ama_con)
# Isolating reviews
ama_con<-ama_con$`Employee Review`

# Making a vector source
ama_con_vec<-VectorSource(ama_con)

#Making a VCorpus
ama_con_corpus<-VCorpus(ama_con_vec)

#Creating Clean_Corpus Function using tm package functions
clean_corpus <- function(corpus){
  corpus <- tm_map(corpus, stripWhitespace)
```

```
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, tolower)
  corpus <- tm_map(corpus, PlainTextDocument)
  corpus <- tm_map(corpus, removeWords, c('amazon','company',stopwords("en")))
          tm_map(corpus, stemDocument, language = "english")
  return(corpus)
}

#For Negative Reviews

#Applying clean_corpus to Amazon Cons reviews corpus
clean_ama_con_corpus<-clean_corpus(ama_con_corpus)

#Creating a bigram tokenizer function
BigramTokenizer <-
  function(x)
    unlist(lapply(ngrams(words(x), 2), paste, collapse = " "), use.names = FALSE)


#Create term-document matrix (TDM) from our amazon cons clean corpus
amazon_con_tdm <-TermDocumentMatrix(clean_ama_con_corpus, control =
list(tokenize=BigramTokenizer))
dim(amazon_con_tdm)

#Convert the amazon_con_tdm to matrix
amazon_con_m<- as.matrix(amazon_con_tdm)
dim(amazon_con_m)
amazon_con_m[1000:1005,6:10]

ama_c_freq <- sort(rowSums(amazon_con_m),decreasing=T)

# Creating a word cloud of 25 negative words in employee reviews
wordcloud(names(ama_c_freq),ama_c_freq,max.words=25,colors=c("grey80","darkgoldenro
d1","tomato"))
```