EDA

Library

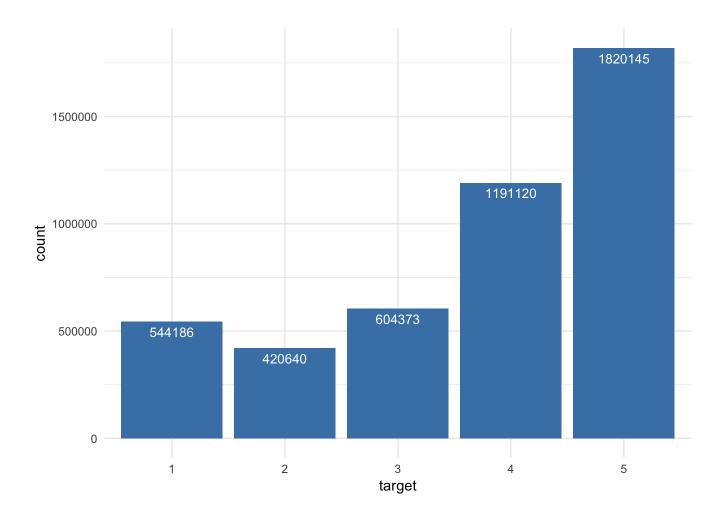
```
if(!require("pacman")) install.packages("pacman")
## Loading required package: pacman
p_load(wordcloud, ggraph, igraph, Rmisc, scales, tidytext, text2vec,
stopwords, Matrix, tokenizers, knitr, keras, tensorflow, magrittr,
tidyverse, caret, flexdashboard, shiny, rmarkdown, Hmisc, DT,
data.table, viridis, leaflet.extras, htmltools, leaflet, jsonlite,
rjson, syuzhet, reticulate, glue, ggpubr)
```

Load Data

```
review <- readRDS(file = "../Data/review_restaurants")
business <- readRDS(file = "../Data/business_restaurants")
sub business <- business[c("city", "business_id", "categories")]</pre>
```

Distribution of Star

```
review %>%
  ggplot(aes(factor(stars))) +
  geom_bar(fill = "steelblue") +
  geom_text(stat = "count", aes(label=..count..), vjust = 1.6, color =
"white", size=3.5) +
  xlab("target") +
  theme minimal()
```

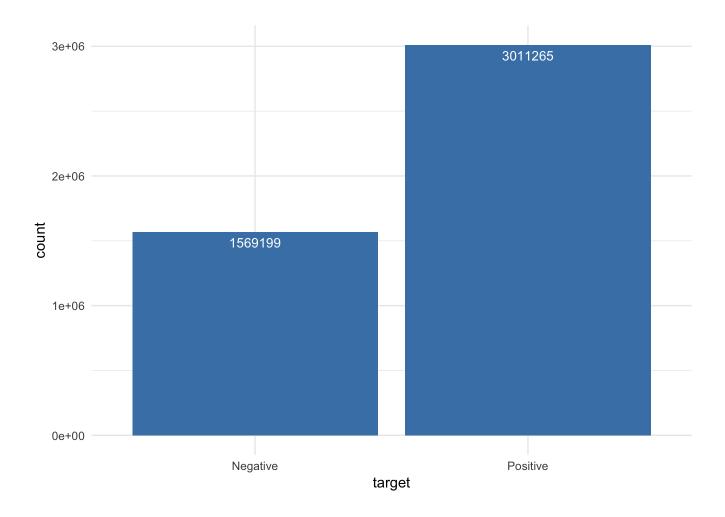


Target Variable

```
review$Target <- ifelse(review$stars > 3, "Positive", "Negative")
newdata <- review[c(1, 3, 8, 10)]
newdata$Target <- as.factor(newdata$Target)
rm(review)
invisible(gc())</pre>
```

Target Distribution

```
newdata %>%
  ggplot(aes(factor(Target))) +
  geom_bar(fill = "steelblue") +
  geom_text(stat = "count", aes(label=..count..), vjust = 1.6, color =
"white", size=3.5) +
  xlab("target") +
  theme_minimal()
```

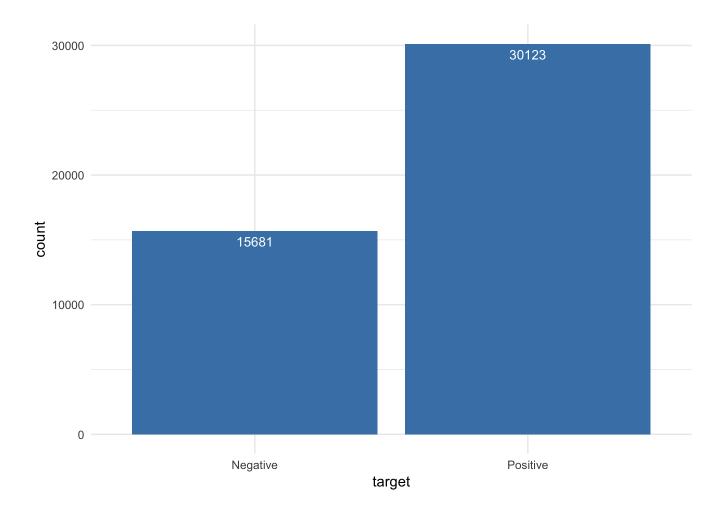


Subset Sample

```
alldata <- newdata[sample(1:nrow(newdata), 0.01 * nrow(newdata),
replace = FALSE), ]
rm(newdata)
invisible(gc())</pre>
```

Sample Target Distribution

```
alldata %>%
  ggplot(aes(factor(Target))) +
  geom_bar(fill = "steelblue") +
  geom_text(stat = "count", aes(label=..count..), vjust = 1.6, color =
"white", size=3.5) +
  xlab("target") +
  theme minimal()
```



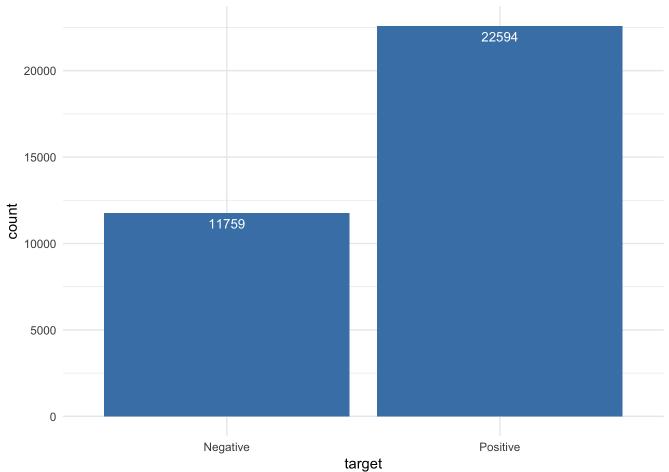
Split into Train and Test and add label

```
smp_size <- floor(0.75 * nrow(alldata))
train_index <- sample(seq_len(nrow(alldata)), size = smp_size)
train <- alldata[train_index,]
test <- alldata[-train_index,]
train$group <- "Train"
test$group <- "Test"
rm(alldata)</pre>
```

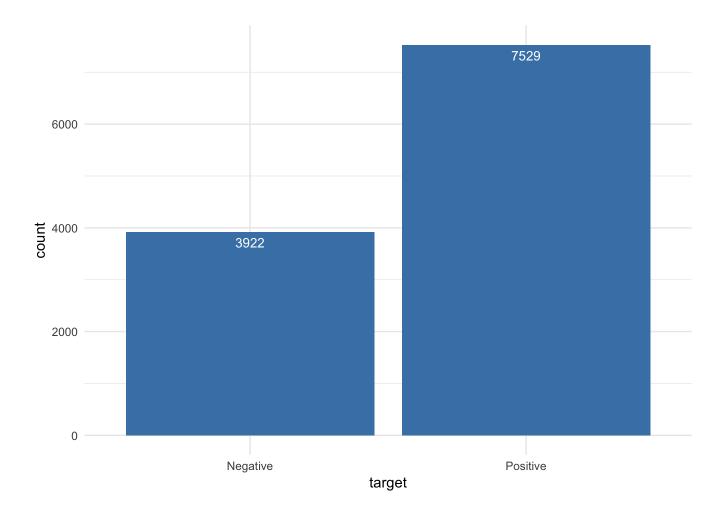
Check Train and Test Distribution

```
train %>%
  ggplot(aes(factor(Target))) +
  geom_bar(fill = "steelblue") +
  geom_text(stat = "count", aes(label=..count..), vjust = 1.6, color =
"white", size=3.5) +
```

```
xlab("target") +
theme_minimal()
```



```
test %>%
  ggplot(aes(factor(Target))) +
  geom_bar(fill = "steelblue") +
  geom_text(stat = "count", aes(label=..count..), vjust = 1.6, color =
"white", size=3.5) +
  xlab("target") +
  theme minimal()
```



Combine Data

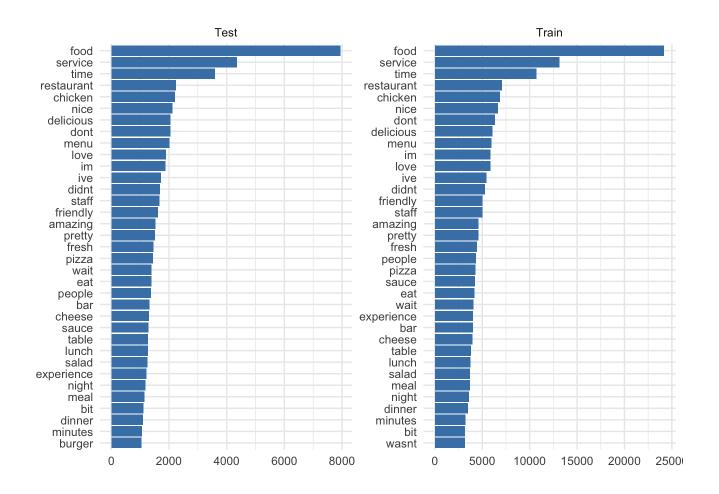
```
alldata <- rbind(train, test)
rm(test)
rm(train)
rm(smp_size)
invisible(gc())</pre>
```

Tokenize the word

```
m_alldata <- merge(sub_business, alldata, "business_id")

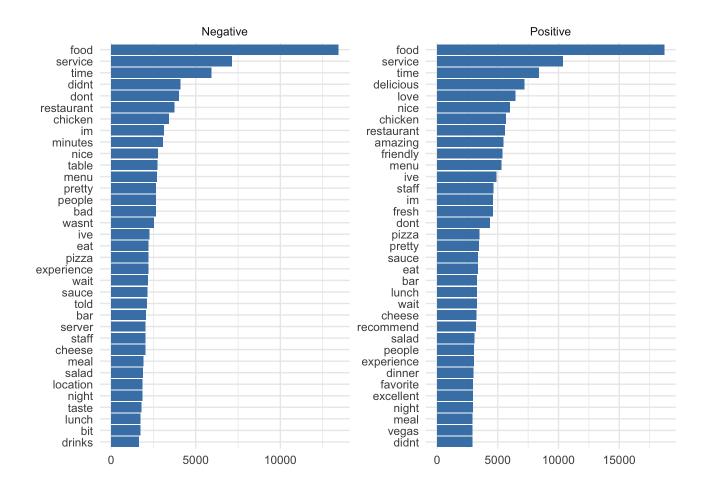
tokens <- m_alldata %>%
   mutate(text = str_replace_all(text, "[^[:alpha:][:space:]]+", ""))
%>%
   unnest tokens(word, text)
```

```
temp <- tokens %>%
  count(word, sort = TRUE) %>%
  top n(10, n)
setDT(temp)
datatable(temp)
tokens %<>%
  anti join(stop words, by = "word")
temp <- tokens %>%
  count(word, sort = TRUE) %>%
  top n(10, n)
setDT(temp)
datatable(temp)
Words for Different Group
scale x reordered <- function(..., sep = " ") {</pre>
  reg <- paste0(sep, ".+$")
  ggplot2::scale x discrete(labels = function(x) gsub(reg, "", x),
}
reorder_within <- function(x, by, within, fun = mean, sep = " ",
...) {
 new x <- paste(x, within, sep = sep)
 stats::reorder(new x, by, FUN = fun)
}
tokens %>%
  select(word, group) %>%
  group by (group) %>%
  count(word, group, sort = TRUE) %>%
  top n(35, n) %>%
  ungroup() %>%
  ggplot(aes(reorder within(word, n, group), n)) +
  geom col(fill = "steelblue") +
  scale x reordered() +
  labs(x = "", y = "") +
  coord flip() +
  theme minimal() +
  facet wrap(~ group, ncol = 2, scales = "free")
```



Words for Different Target

```
tokens %>%
  select(word, Target) %>%
  group_by(Target) %>%
  count(word, sort = TRUE) %>%
  top_n(35, n) %>%
  ungroup() %>%
  ggplot(aes(reorder_within(word, n, Target), n)) +
  geom_col(fill = "steelblue") +
  scale_x_reordered() +
  labs(x = "", y = "") +
  coord_flip() +
  theme_minimal() +
  facet_wrap(~ Target, ncol = 2, scales = "free")
```



Frequency of the words

```
tokens %>%
 group by (Target) %>%
 count(word, sort = TRUE) %>%
  left_join(tokens %>%
              group by (Target) %>%
              summarise(total = n()), by = "Target") %>%
 mutate(freq = n/total) %>%
  select(Target, word, freq) %>%
  spread(Target, freq) %>%
  arrange(`Positive`, `Negative`) %>%
  ggplot(aes(`Positive`, `Negative`)) +
  geom jitter(alpha = 0.05, size = 0.5, width = 0.25, height = 0.25) +
  geom_abline(color = "red") +
  geom text(aes(label = word), check overlap = TRUE, vjust = 1.5) +
  scale x log10(labels = percent format()) +
  scale y log10(labels = percent format()) +
```

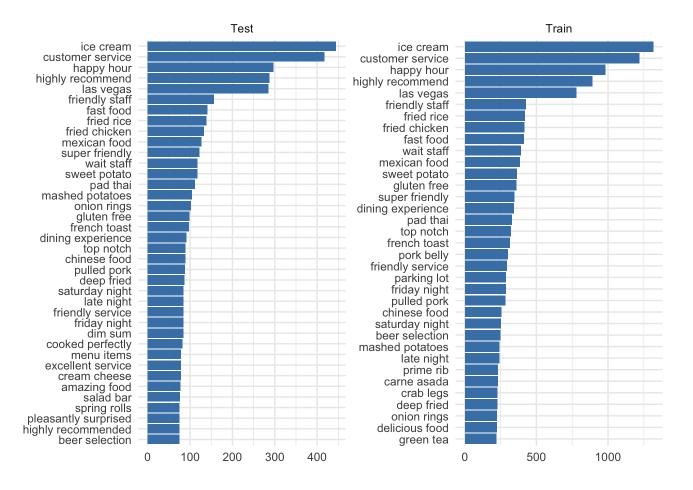
```
labs(x = "Positive", y = "Negative") +
theme minimal()
                                                                                                                                                                                                         food
  1.000%
                                                                                                                                                                                           time
                                                                                                                                                                        didnt
                                                                                                                                                      minutes
                                                                                                                                                   told server staff
                                                                                                                                        manager hourday fresh
                                                                                     worst bland bill pay star feel de poor paid fine sit tacos ama: awful understand ate stop excellent barely tiny type loved gross hair raw grill wow nasty sucks refill fix pool lol fantastic
  0.100%
                                                                                                                                                                                   delicious
                                                                                                                                                                               amazing
                      nasty sucks refill fix pool lol fantastic unprofessional ruined tells fee til heyboy rudely gratuity nopeknife hill bits bf à valley inattentive spit tone kill dude cuts gotta argued unimpressed hated hrs mid jar dj ahi art drainreported ir pf caland lil killer deposit lids hostile proof ad cdr del meltsterrific defensive ig annicht bb fort hints aggravation allotaoli fifty erlion sffoster accuse dat jahpbillvea dt allatar
                                                                           tasteless
  0.010%
  0.001%
                             accuse dat jahpbillyea dt allatar
abort alsalløbrettsolceskhalisa
                          absolutley/abrsbwwdw bitina
                             abbeyalize iii bolapte bipup zesty
  0.000%
                               0.000%
                                                                       0.001%
                                                                                                               0.010%
                                                                                                                                                      0.100%
                                                                                                                                                                                              1.000%
```

Positive

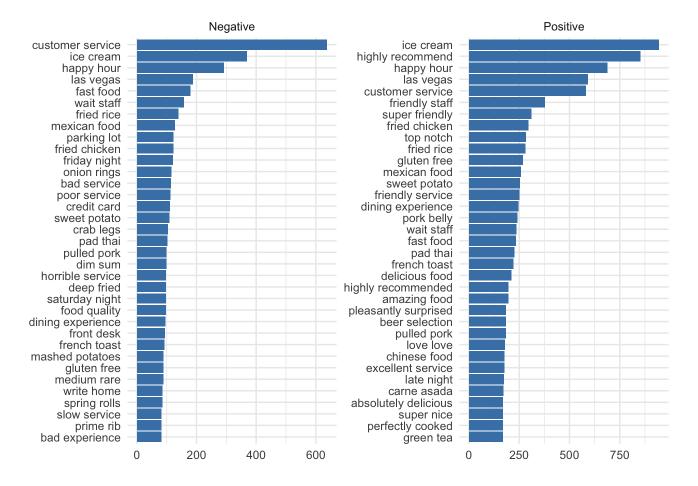
Tokens Bigrams

```
unite(bigram, word1, word2, sep = " ")
temp <- bigrams %>%
  count(bigram, sort = TRUE) %>%
  top n(10, n)
setDT(temp)
datatable(temp)
Bigrams graphs
```

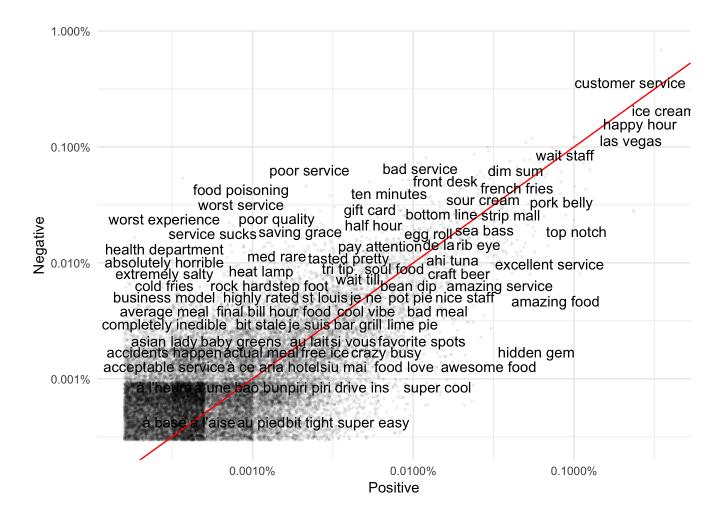
```
bigrams %>%
 select(bigram, group) %>%
 group by(group) %>%
 count(bigram, group, sort = TRUE) %>%
  top n(35, n) %>%
 ungroup() %>%
 ggplot(aes(reorder_within(bigram, n, group), n)) +
 geom col(fill = "steelblue") +
 scale x reordered() +
  labs(x = "", y = "") +
 coord flip() +
  theme minimal() +
  facet_wrap(~ group, ncol = 2, scales = "free")
```



```
bigrams %>%
  filter(group == "Train") %>%
  select(bigram, Target) %>%
  group_by(Target) %>%
  count(bigram, sort = TRUE) %>%
  top_n(35, n) %>%
  ungroup() %>%
  ggplot(aes(reorder_within(bigram, n, Target), n)) +
  geom_col(fill = "steelblue") +
  scale_x_reordered() +
  labs(x = "", y = "") +
  coord_flip() +
  theme_minimal() +
  facet wrap(~ Target, ncol = 2, scales = "free")
```



```
bigrams %>%
  group by (Target) %>%
  count(bigram, sort = TRUE) %>%
  left join(bigrams %>%
              group by (Target) %>%
              summarise(total = n()), by = "Target") %>%
 mutate(freg = n/total) %>%
  select(Target, bigram, freq) %>%
  spread(Target, freq) %>%
  arrange(`Positive`, `Negative`) %>%
  ggplot(aes(`Positive`, `Negative`)) +
  geom jitter(alpha = 0.05, size = 0.5, width = 0.25, height = 0.25) +
  geom abline(color = "red") +
  geom text(aes(label = bigram), check overlap = TRUE, vjust = 1.5) +
  scale x log10(labels = percent format()) +
  scale y log10(labels = percent format()) +
  labs(x = "Positive", y = "Negative") +
  theme minimal()
```



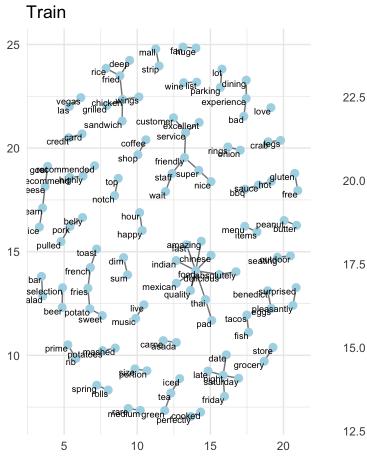
Network of Bigrams (Data)

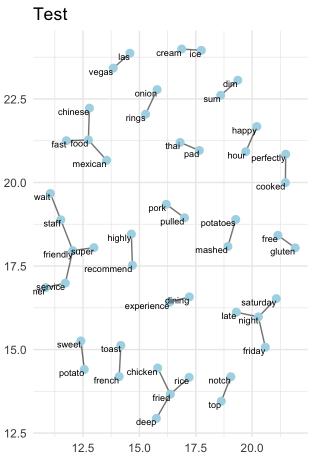
```
p1 <- bigrams %>%
  filter(group == "Train") %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  count(word1, word2, sort = TRUE) %>%
  filter(n > 150) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = 0.8), show.legend = FALSE) +
  geom_node_point(color = "lightblue", size = 2.5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1, size = 2.4)
+
  labs(x = "", y = "") +
  ggtitle("Train") +
  theme_minimal()

p2 <- bigrams %>%
```

```
filter(group == "Test") %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  count(word1, word2, sort = TRUE) %>%
  filter(n > 80) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = 0.8), show.legend = FALSE) +
  geom_node_point(color = "lightblue", size = 2.5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1, size = 2.4)
+
  labs(x = "", y = "") +
  ggtitle("Test") +
  theme_minimal()

multiplot(p1, p2, cols = 2)
  Train  Test
```

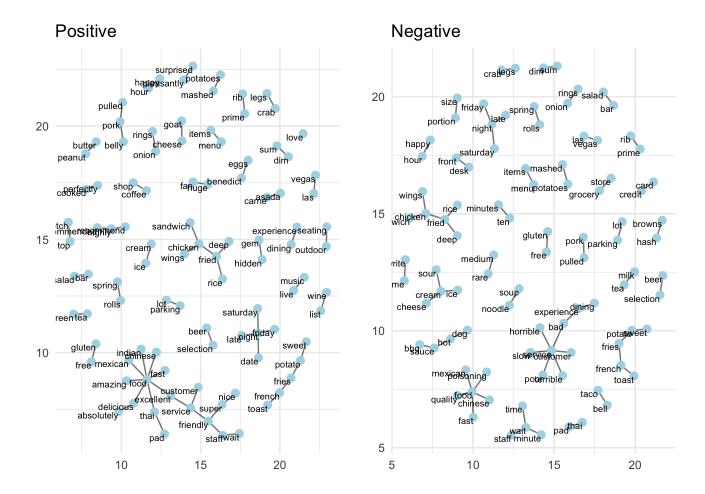




Network of Bigrams (Group)

p1 <- bigrams %>%

```
filter(Target == "Positive") %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  count(word1, word2, sort = TRUE) %>%
  filter(n > 150) %>%
  graph from data frame() %>%
  ggraph(layout = "fr") +
  geom edge link(aes(edge alpha = 0.8), show.legend = FALSE) +
  geom node point(color = "lightblue", size = 2.5) +
  geom node text(aes(label = name), vjust = 1, hjust = 1, size = 2.4)
  labs(x = "", y = "") +
  ggtitle("Positive") +
  theme_minimal()
p2 <- bigrams %>%
  filter(Target == "Negative") %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  count(word1, word2, sort = TRUE) %>%
  filter(n > 80) %>%
  graph from data frame() %>%
  ggraph(layout = "fr") +
  geom edge link(aes(edge alpha = 0.8), show.legend = FALSE) +
  geom node point(color = "lightblue", size = 2.5) +
  geom node text(aes(label = name), vjust = 1, hjust = 1, size = 2.4)
  labs(x = "", y = "") +
  qqtitle("Negative") +
  theme minimal()
multiplot(p1, p2, cols = 2)
```



Sentiment Analysis

sentiments %>%
 sample_n(10) %>%
 kable()

word	sentiment	lexicon	score
unfulfilled	negative	bing	NA
cessation	negative	nrc	NA
defy	negative	nrc	NA
deportation	sadness	nrc	NA
pertinacious	negative	bing	NA
fortune	positive	nrc	NA
perversion	negative	nrc	NA
doubt	sadness	nrc	NA
deadlocking	negative	loughran	NA
retort	negative	nrc	NA

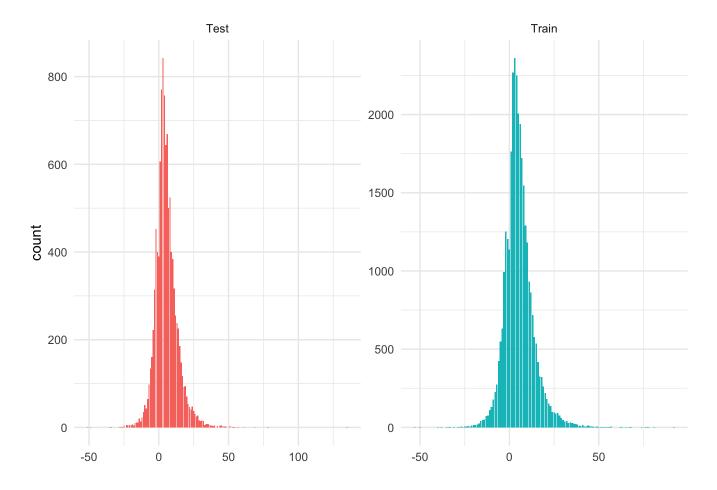
sentiments %>%

```
filter(lexicon == "AFINN") %>%
sample_n(10) %>%
kable()
```

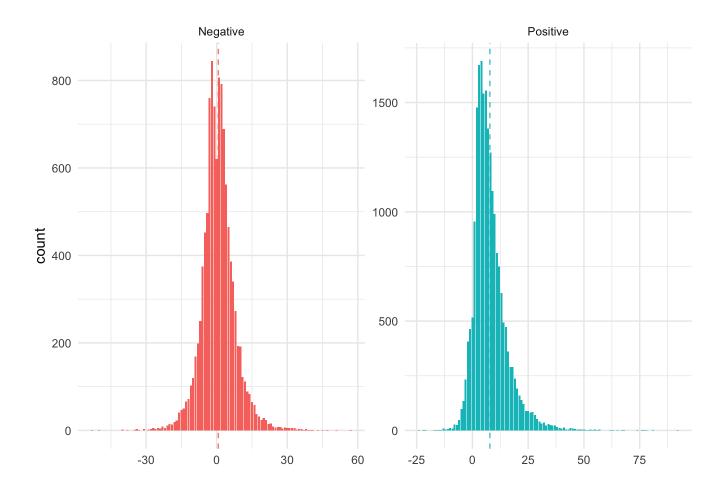
word	sentiment	lexicon	score
burdens	NA	AFINN	-2
bitches	NA	AFINN	-5
discontented	NA	AFINN	-2
praise	NA	AFINN	3
legally	NA	AFINN	1
stupidly	NA	AFINN	-2
demand	NA	AFINN	-1
favorited	NA	AFINN	2
waste	NA	AFINN	-1
misbehaves	NA	AFINN	-2

Sentiment Score for alldata

```
tokens_sent <- inner_join(tokens, get_sentiments("afinn"))
## Joining, by = "word"
tokens_sent %>%
  group_by(review_id, Target, group) %>%
  summarise(score = sum(score)) %>%
  ungroup() %>%
  ggplot(aes(score, fill = group)) +
  geom_bar(show.legend = FALSE) +
  labs(x = "") +
  facet_wrap(~ group, ncol = 2, scales = "free") +
  theme minimal()
```



```
tokens_sent %>%
  filter(group == "Train") %>%
  group_by(review_id, Target) %>%
  summarise(score = sum(score)) %>%
  ungroup() %>%
  group_by(Target) %>%
  mutate(avg = mean(score)) %>%
  ungroup() %>%
  ggplot(aes(score, fill = factor(Target))) +
  geom_bar(show.legend = FALSE) +
  geom_vline(aes(xintercept = avg, colour = factor(Target)), linetype
= "dashed", size = 0.4, show.legend = FALSE) +
  labs(x = "") +
  facet_wrap(~ Target, ncol = 2, scales = "free") +
  theme minimal()
```



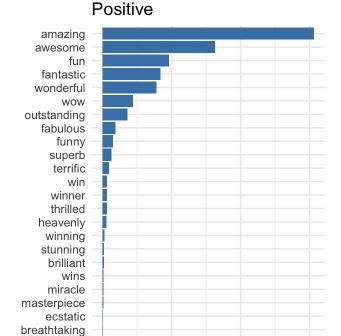
Most common words

```
p1 <- tokens_sent %>%
  select(word, score) %>%
  add_count(word) %>%
  distinct() %>%
  arrange(desc(score), desc(n)) %>%
  slice(1:30) %>%
  ggplot(aes(reorder_within(word, n, score), n)) +
  geom_col(fill = "steelblue", show.legend = FALSE) +
  scale_x_reordered() +
  labs(x = "", y = "") +
  ggtitle("Positive") +
  coord_flip() +
  theme_minimal()

p2 <- tokens_sent %>%
  select(word, score) %>%
```

```
add_count(word) %>%
distinct() %>%
arrange(score, desc(n)) %>%
slice(1:30) %>%
ggplot(aes(reorder_within(word, n, score), n)) +
geom_col(fill = "steelblue", show.legend = FALSE) +
scale_x_reordered() +
labs(x = "", y = "") +
ggtitle("Negative") +
coord_flip() +
theme_minimal()
```

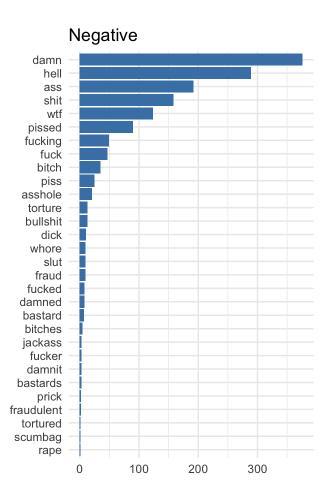
multiplot(p1, p2, cols = 2)



2000

4000

6000



Wordcloud (Group)

Imao overjoyed

godsend

euphoric

exuberant

rejoice

hurrah

```
counts <- tokens %>%
  filter(Target == "Positive") %>%
  count(word, sort = TRUE) %>%
```

```
top_n(100, n)
wordcloud(counts$word, counts$n, random.order = FALSE, colors =
RColorBrewer::brewer.pal(8,"Dark2"))
```

```
dishes flavor bread
beermeat breakfast
burger
                                                                          top experience people pricewasnt
                                            pork sweet dinner ricelunch salad fried souplot cheese  ricelunch salad fried restaurant shrimp
                                                                                                                                                                                                                                                        sauce location
                        visit amazing
                                                                                                                                                                                                                                                                                                                   loved
                                                                                                                                                                                                                                                                                             pretty
                                                                                                                                                                                                                                                                                                                                           ďrink
                                                                                                                                                                                                                                                                                                                        excellent
                                                                                                                                                                                                                                                                           fresh
                                                                                                                                                                                                                                                                                                                           atmosphere at mosphere at mosp
                feel meal ive
                                 times
                                          vegas im
selection fries eat
       friends favorite Sedish drinks
                  steakday
                                                       spicy perfect pizza wait tasty coffee hugeworth didnt supertable
                                           highly hugeworthdidnt supertable server
                                                                    spot awesome sushi coming options quality taste sandwich special enjoyed inside
```

```
counts <- tokens %>%
  filter(Target == "Negative") %>%
  count(word, sort = TRUE) %>%
  top_n(100, n)
wordcloud(counts$word, counts$n, random.order = FALSE, colors =
RColorBrewer::brewer.pal(8,"Dark2"))
```

```
sweet prices
                              bread
          sandwich decent
                            disappointed
      visit half burger price rice served
        quality lunch experience soup
   looked drink pretty minutes meatfriends
flavor salad nice restaurant friendly special people times you
                              times youre tasted
  fried table Sel
friend ive
                                 meal cold
                                 bar waiting spinally
  worth eat hot pizza
                                o o o money
 checkwaitress dinner love to water manager customer
             tables cust
```

Wordcloud Bigram

```
counts <- bigrams %>%
  filter(Target == "Positive") %>%
  count(bigram, sort = TRUE) %>%
  top_n(100, n)
wordcloud(counts$bigram, counts$n, random.order = FALSE, colors =
RColorBrewer::brewer.pal(8,"Dark2"))
```

```
reasonable prices
                   indian food
                                 medium rare
      saturday night beer selection chicken sandwich
                       amazing tood excellent food
  cooked perfectly
    goat cheese prime rib delicious food caesar salad
                   wine list friendly service
   coffee shop o
                        top notch pork belly strip mall
  menu items
  peanut butter-
                   Φ
                      super friendly
   friday night
 carne asada
                   ਰ frienc
 chinese food
                                               love love
               Œ
  milk tea
late night
                                                 fish tacos
huge fan
parking lot
                                                 onion rings
 green tea
                                                live music 5
                                            chicken salad
                                                         g
                                                         Ε
                                               deep fried 🚡
  bar
                                               date night #
  ad
excellent service mexican food
                                                   hot sauce
                                      thai food
             sweet potatofrench toastcrab legs
 fast service
chicken wings
                                      absolutely delicious
eggs benedict dining experience
                                         french fries
absolutely love highly recommended mashed potatoes
portion size pleasantly surprised perfectly cooked
                            iced tea
     food service taste buds
                                        nice touch
```

```
counts <- bigrams %>%
  filter(Target == "Negative") %>%
  count(bigram, sort = TRUE) %>%
  top_n(100, n)
wordcloud(counts$bigram, counts$n, random.order = FALSE, colors =
RColorBrewer::brewer.pal(8,"Dark2"))
```

```
food arrived hot sauce pork belly gluten free deep fried sports bar noodle soup chinese food mashed potatoes lunch time potato fries crab legs credit card medium rare worst service soy sauce fried chicken horrible service taco bell hash browns write home write home front desk of sold service and sold service terrible service of the se
```

Model Building

Remove Punctuations

```
puncts <- paste(puncts, collapse = "|")</pre>
puncts <- paste("([", puncts, "])", sep = "", collapse = "")</pre>
alldata %<>%
  mutate(text = str replace all(text, "[0-9]{5,}", "#####"),
         text = str replace all(text, "[0-9]{4}", "####"),
         text = str replace all(text, "[0-9]{3}", "###"),
         text = str replace all(text, "[0-9]{2}", "##"),
         text = str replace all(text, puncts, " \\1 "))
Tokenizer
maxlen <- 80
max words <- 10000
emb dim <- 300
tokenizer <- text tokenizer(num words = max words) %>%
  fit text tokenizer(alldata$text)
word idx <- tokenizer$word index</pre>
sequences <- texts to sequences(tokenizer, alldata$text) %>%
  pad sequences(maxlen = maxlen)
invisible(gc())
Split Data
y <- alldata %>% filter(group == "Train") %$% Target
val <- caret::createDataPartition(y, p = 0.15, list = F) %>% c()
X tr <- sequences[train index, ][-val, ]</pre>
y_tr <- y[-val]</pre>
X val <- sequences[train index, ][val, ]</pre>
y val <- y[val]</pre>
X te <- sequences[-train index, ]</pre>
y te <- alldata %>% filter(group == "Test") %$% Target
y tr <- as.numeric(y tr) - 1
y val <- as.numeric(y val) - 1
y te <- as.numeric(y te) - 1
rm(sequences)
invisible(gc())
```

LSTM

```
max features <- 10000
maxlen <- 80
batch size <- 32
embedding dims <- 50
filters <- 64
kernel size <- 5
hidden dims <- 50
model lstm <- keras model sequential()</pre>
model lstm %>%
  layer embedding(input dim = max features, output dim = 128) %>%
  layer lstm(units = 64, dropout = 0.2, recurrent dropout = 0.2) %>%
  layer dense(units = 1, activation = 'sigmoid')
model lstm %>% compile(
  loss = 'binary crossentropy',
 optimizer = 'adam',
 metrics = c('accuracy')
)
model lstm %>% fit(
 X tr, y tr,
 batch size = batch size,
 epochs = 2,
 validation data = list(X val, y val)
results <- model lstm %>% evaluate(
 X te, y te,
 batch size = batch size
setDT(results)
rownames(results) <- "Result"</pre>
datatable(round(results, 3))
```

Assumptions

We assumed 1% of the data and the data provided by yelp can have a great representation of all the restaurants in Canada and America. This is reasonable since most people will react in a same way towards the restaurants they like and they hate. We obtained a really good result from our last model with an accuracy of 87% and we have tested it using randomly collected reviews online from yelp and its accuracy was proven to be right. So we could say our

assumption is reasonable.

Limitations and Uncertainties

In our project, one of the most significant limitations will be the hardware problem. Our computers are not able to run the full 100% data. It will crash due to the memory is not big enough. We will have to sample only 1% the data from the original dataset. Using this 1% data, it still takes a long time to run the code and the computers' temperature increased dramatically. Although the sample size is only 45k, the number of rows of the tokenizer for unigram still exceeded 1 million rows. Since this is only a sampel, there might be some bias and we will not able to collect all the information. if we could have a high performance computer, we would hace a more accurate prediction for the model. With 1% of the data, we are able to achieve an accuracy of 87% so with 100% of the data, I believe we could have an accuracy that will be higher than 97%. Another limitation will be the provided review dataset only has restaurants in several cities. There is no big cities like LA and NYC and no small cities such as Reno and Champaign. So it might not represent all of the restaurants for Canada and America. We should have a larger datasets with restaurants all over the place.

Areas of Future Investigation

In the future, we could have a lot more to do with the current datasets. We are only able to input the restaurant categories for the wordcloud. We might be able to make an app and users can input the restaurant name and see the most used words for that restaurant using unigram, bigram and trigram so they are able to have a glimpse about the restaurant's condition and their popular food. We could also incorporate some other languages when training our neural network model which was misclassified and shown in the presentation slide.