EDA

Library

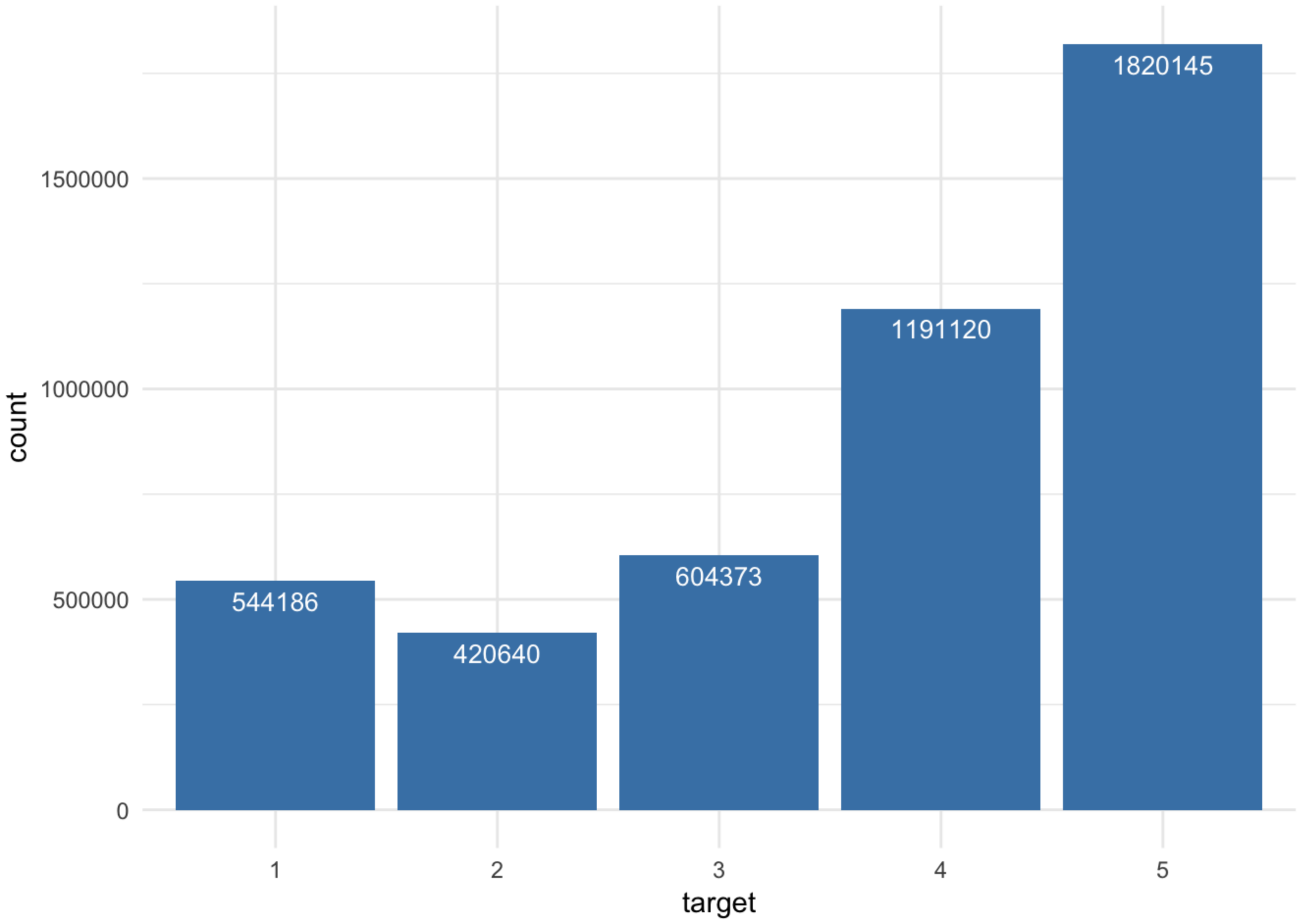
```
if(!require("pacman")) install.packages("pacman")
## Loading required package: pacman
p_load(wordcloud, ggraph, igraph, Rmisc, scales, tidytext, text2vec,
stopwords, Matrix, tokenizers, knitr, keras, tensorflow, magrittr,
tidyverse, caret, flexdashboard, shiny, rmarkdown, Hmisc, DT,
data.table, viridis,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

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()
```

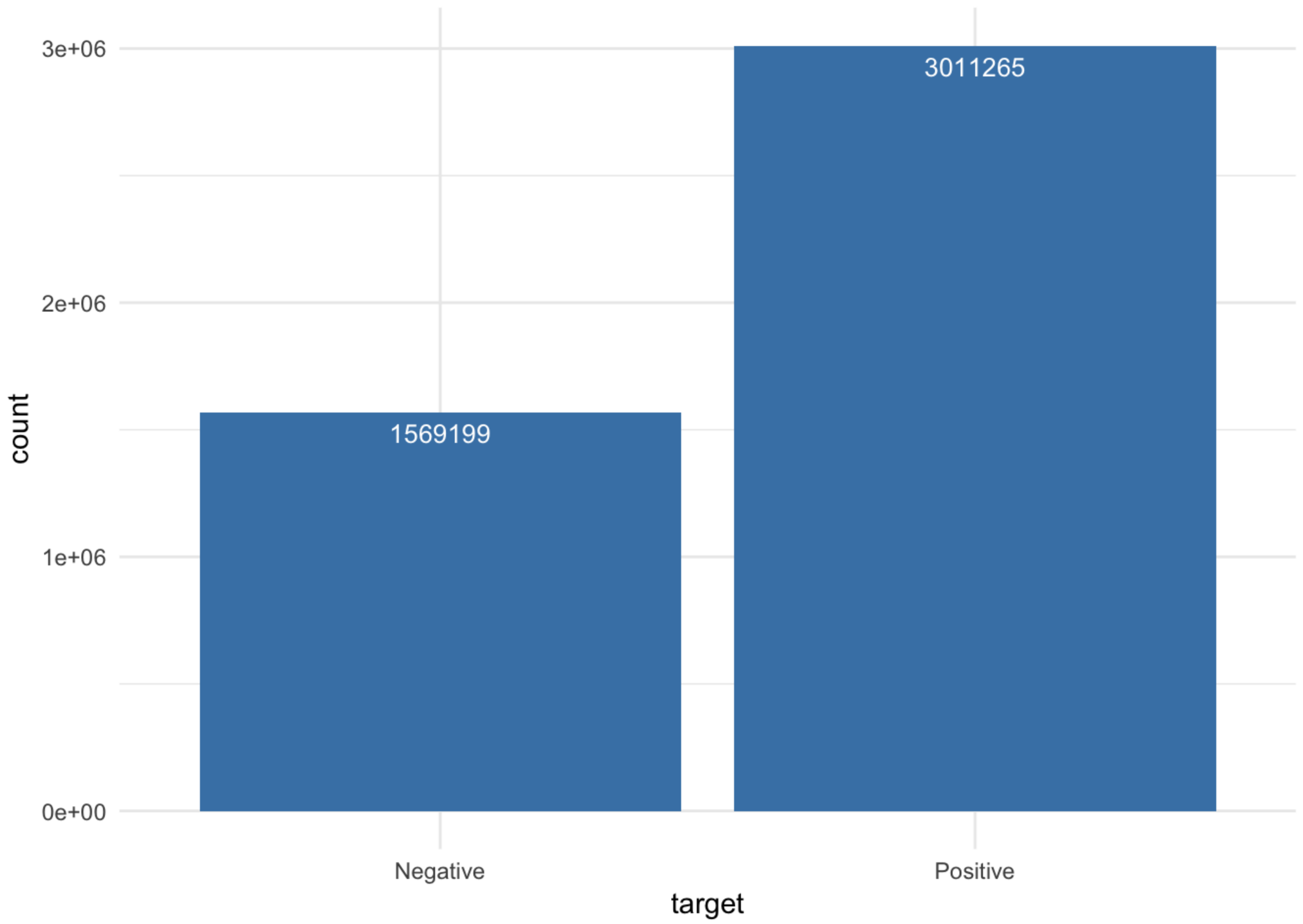


Target Variable

```
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())
```

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()
```

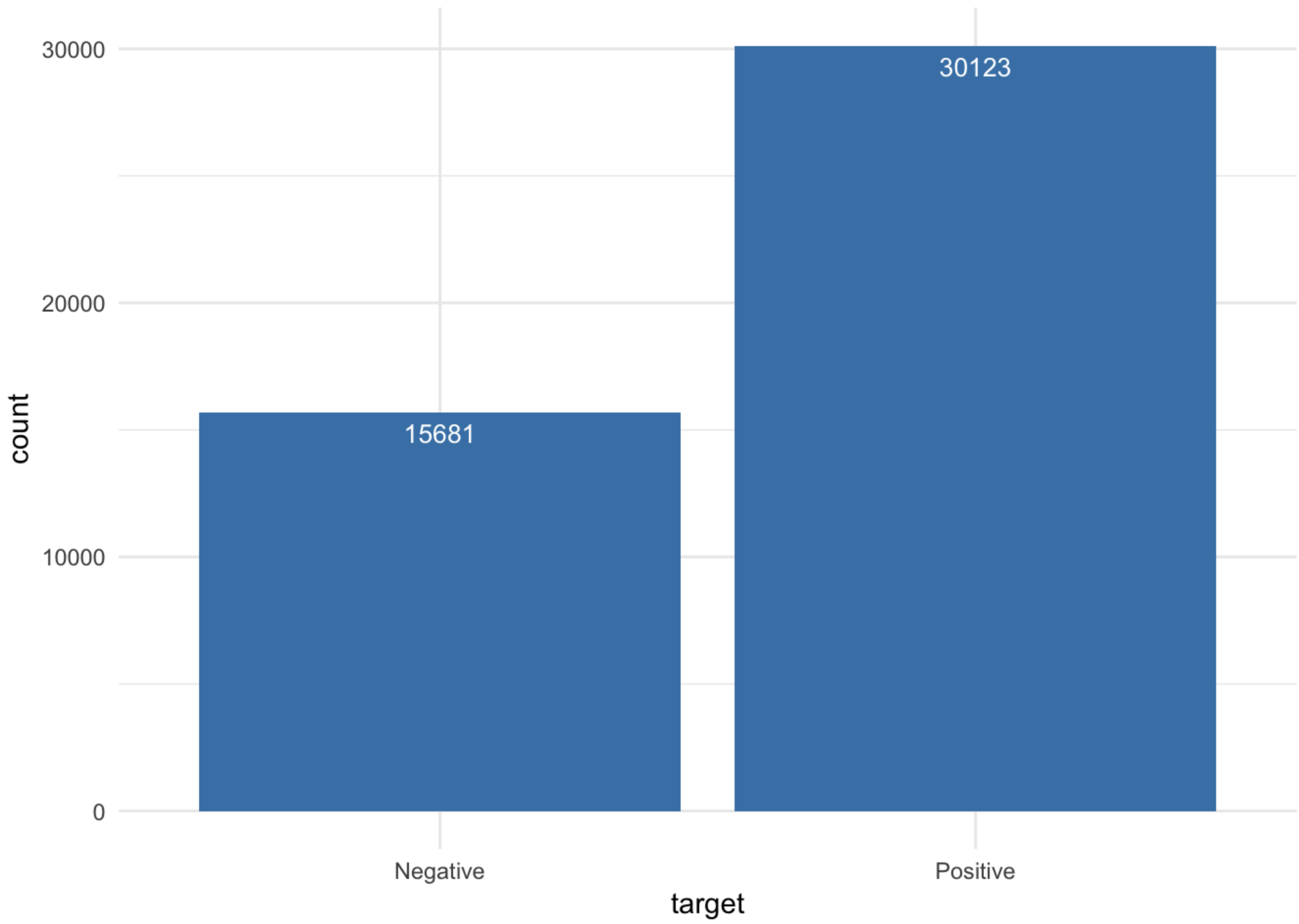


Subset Sample

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

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()
```

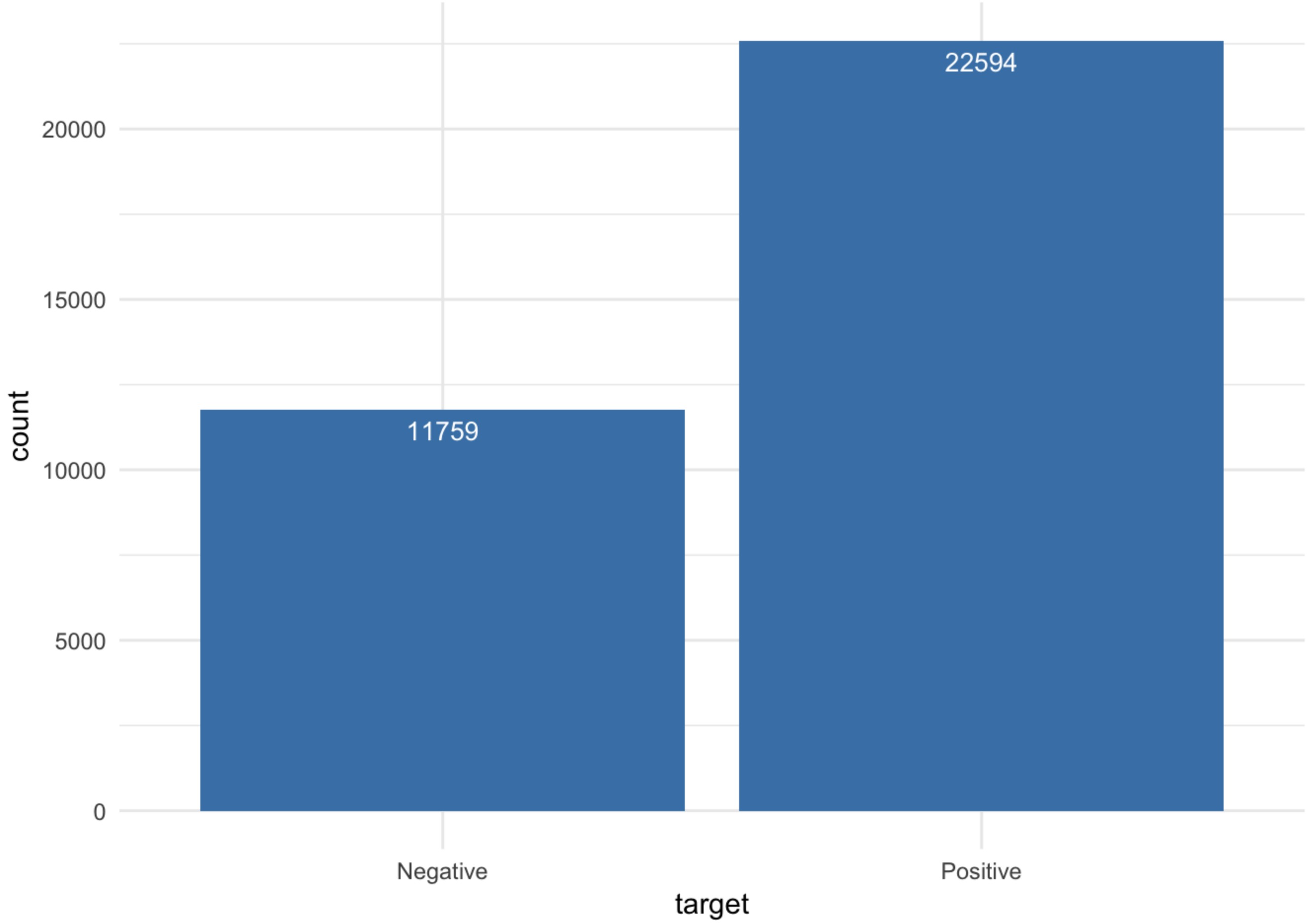


Split into Train and Test and add label

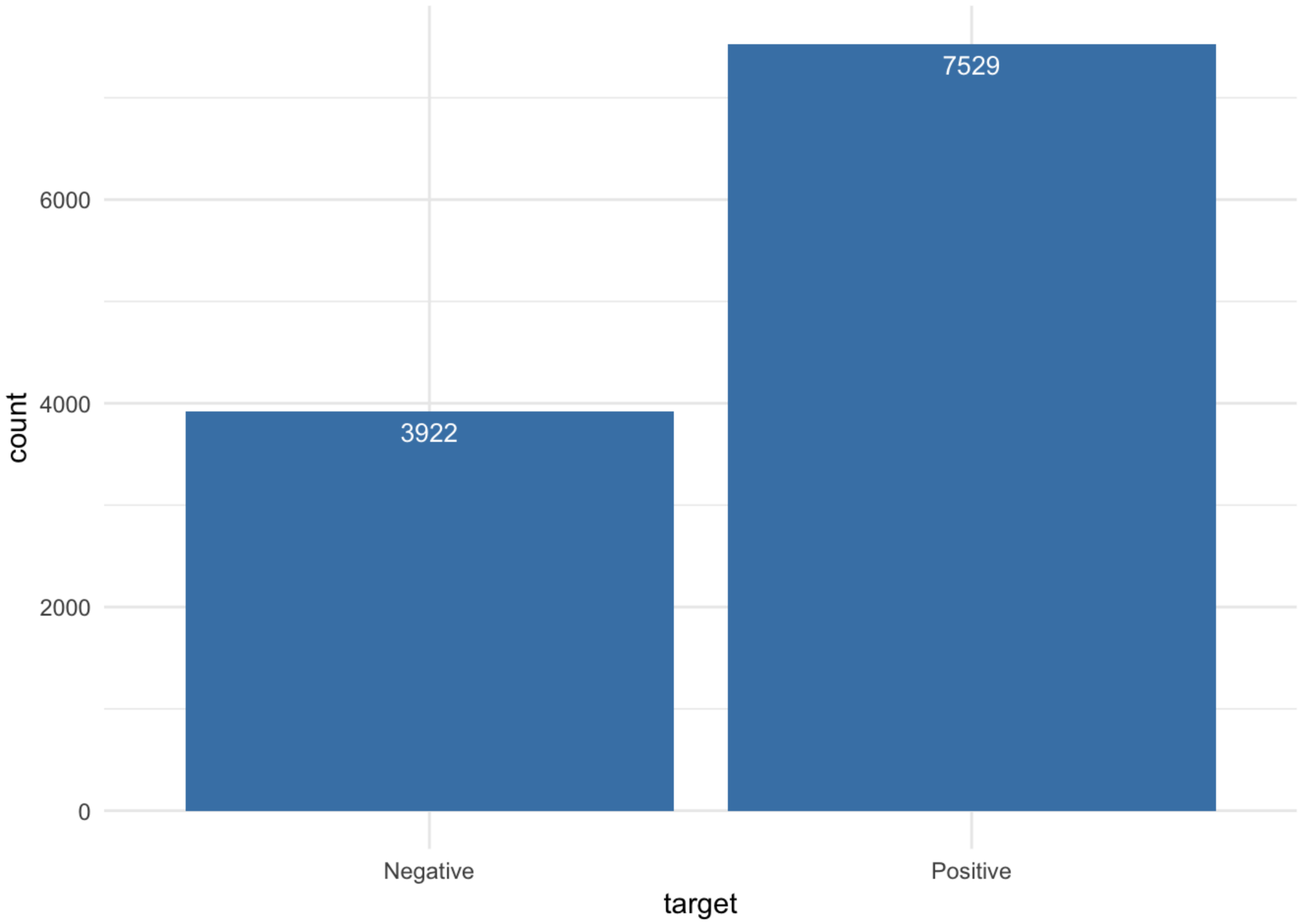
```
smp_size <- floor(0.75 * nrow(alldata))
train_index <- sample(seq_len(nrow(alldata)), size = smp_size)
train <- alldata[train_index,]
test <- alldata[-train_index,]
train$group <- "Train"
test$group <- "Test"
rm(alldata)
```

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())
```

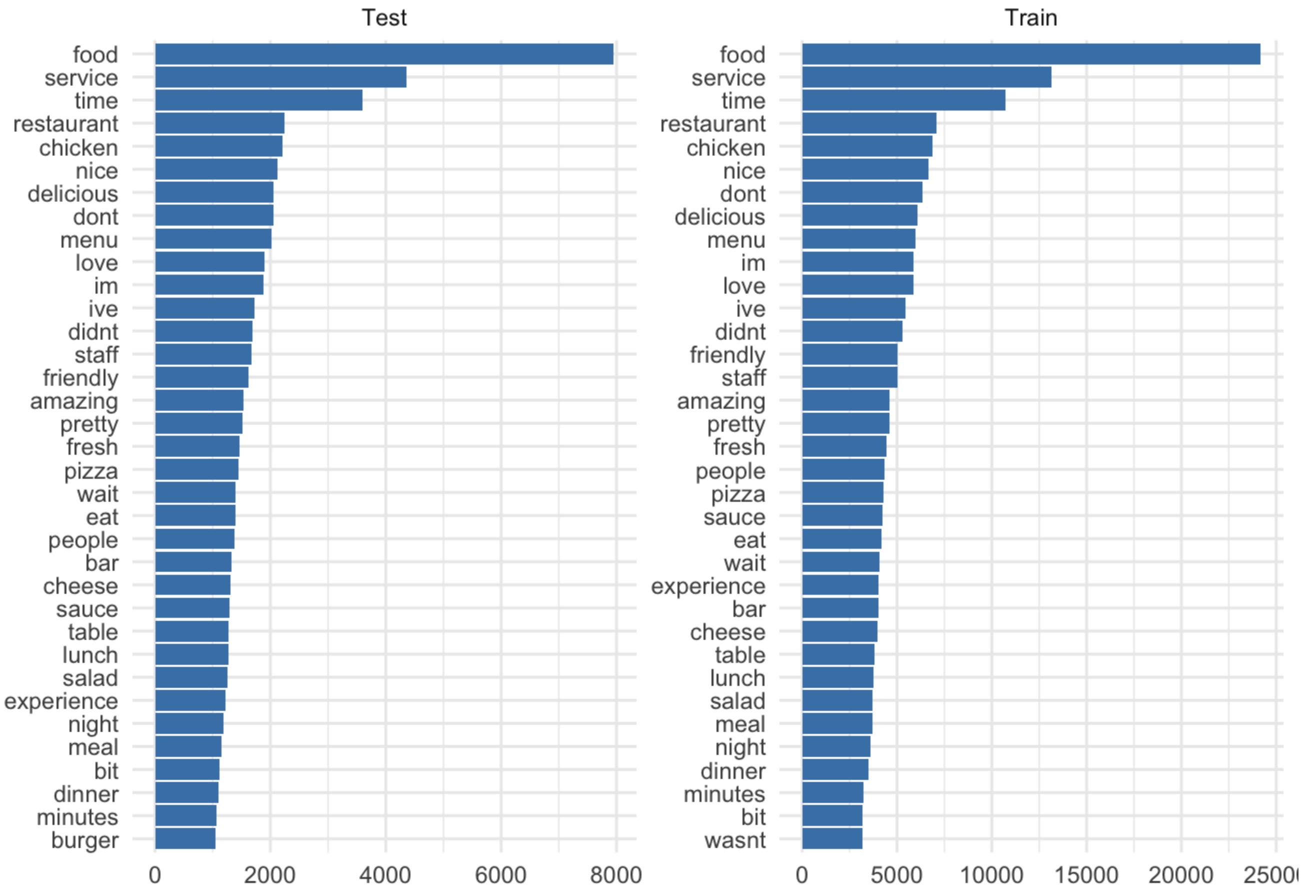
Tokenize the word

```
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)
```

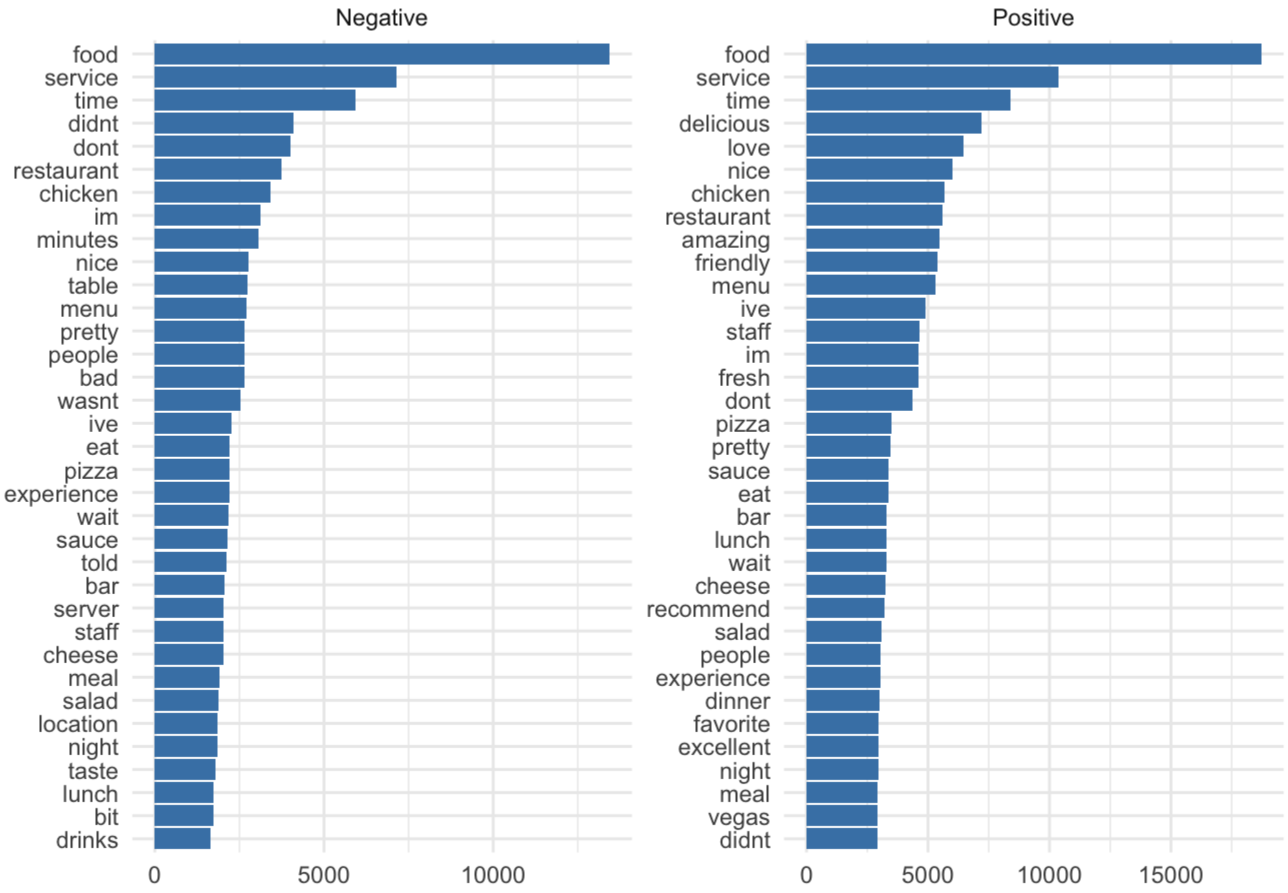
Words for Different Group

```
scale_x_reordered <- function(..., sep = "___") {
  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")
```



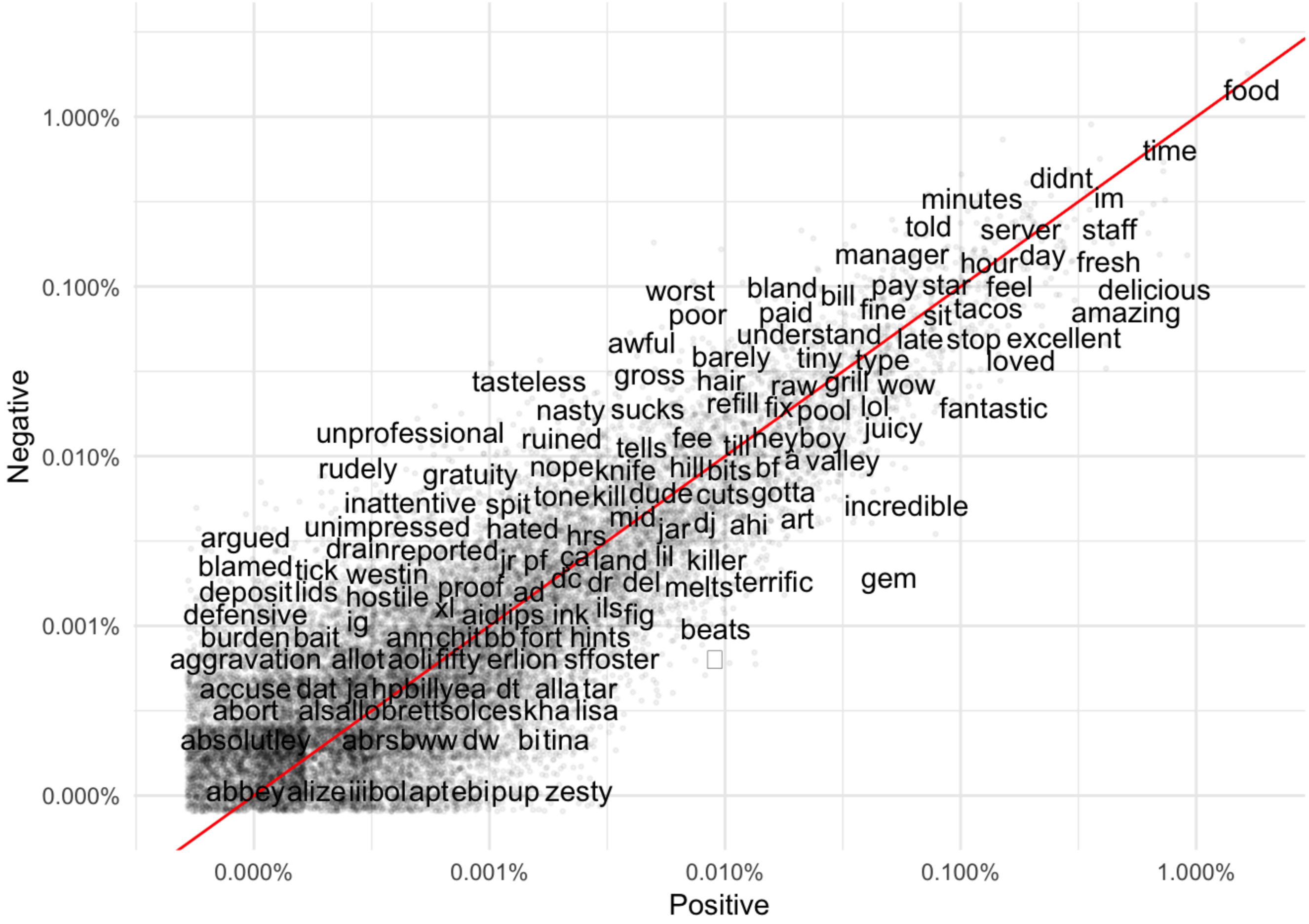
Words for Different Target

```
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()
```

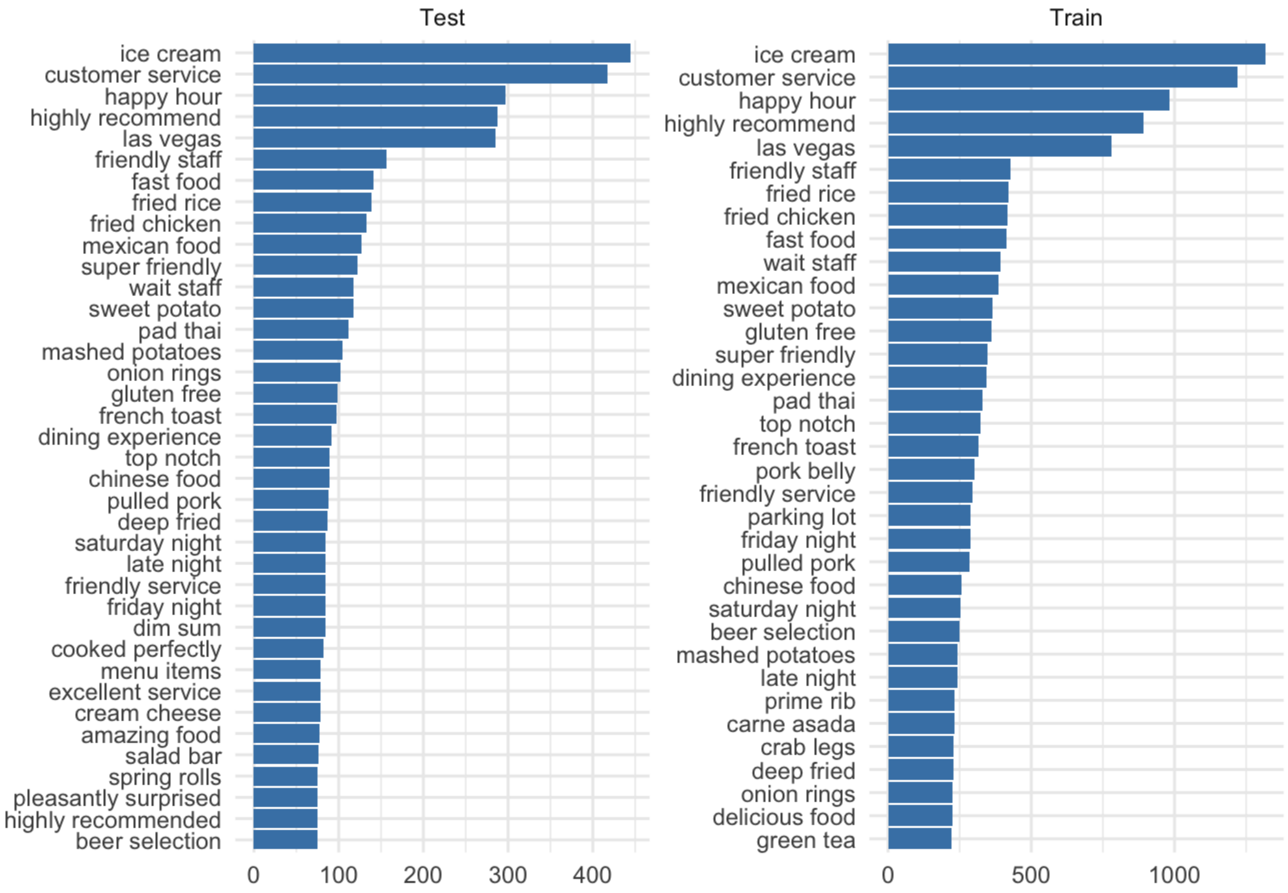


Tokens Bigrams

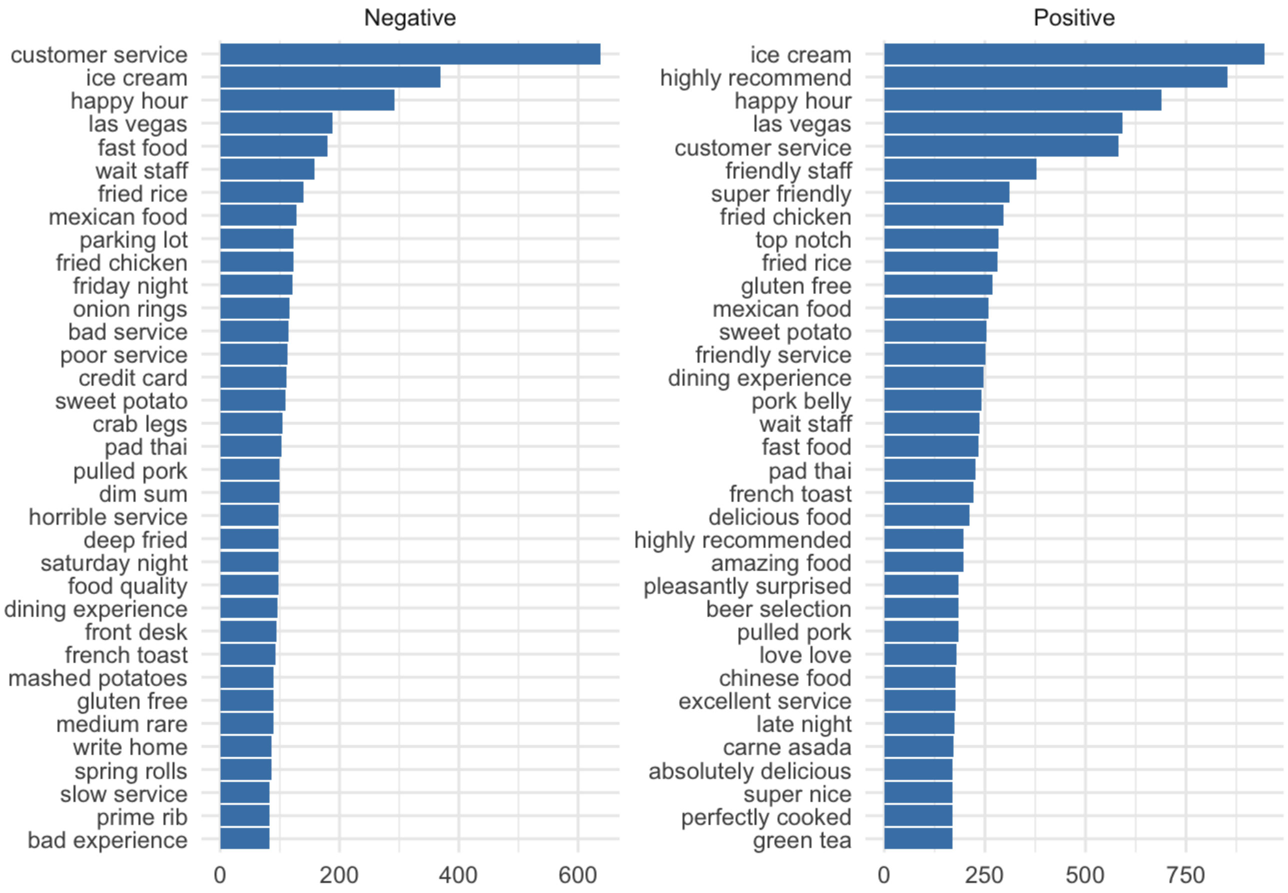
```
bigrams <- m_alldata %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

temp <- bigrams %>%
  count(bigram, sort = TRUE) %>%
  top_n(10, n)
setDT(temp)
datatable(temp)
bigrams %<>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word,
         !word2 %in% stop_words$word,
         !str_detect(word1, "[[:digit:]]"),
         !str_detect(word2, "[[:digit:]]")) %>%
  unite(bigram, word1, word2, sep = " ")


temp <- bigrams %>%
  count(bigram, sort = TRUE) %>%
  top_n(10, n)
setDT(temp)
datatable(temp)
```

Bigrams graphs

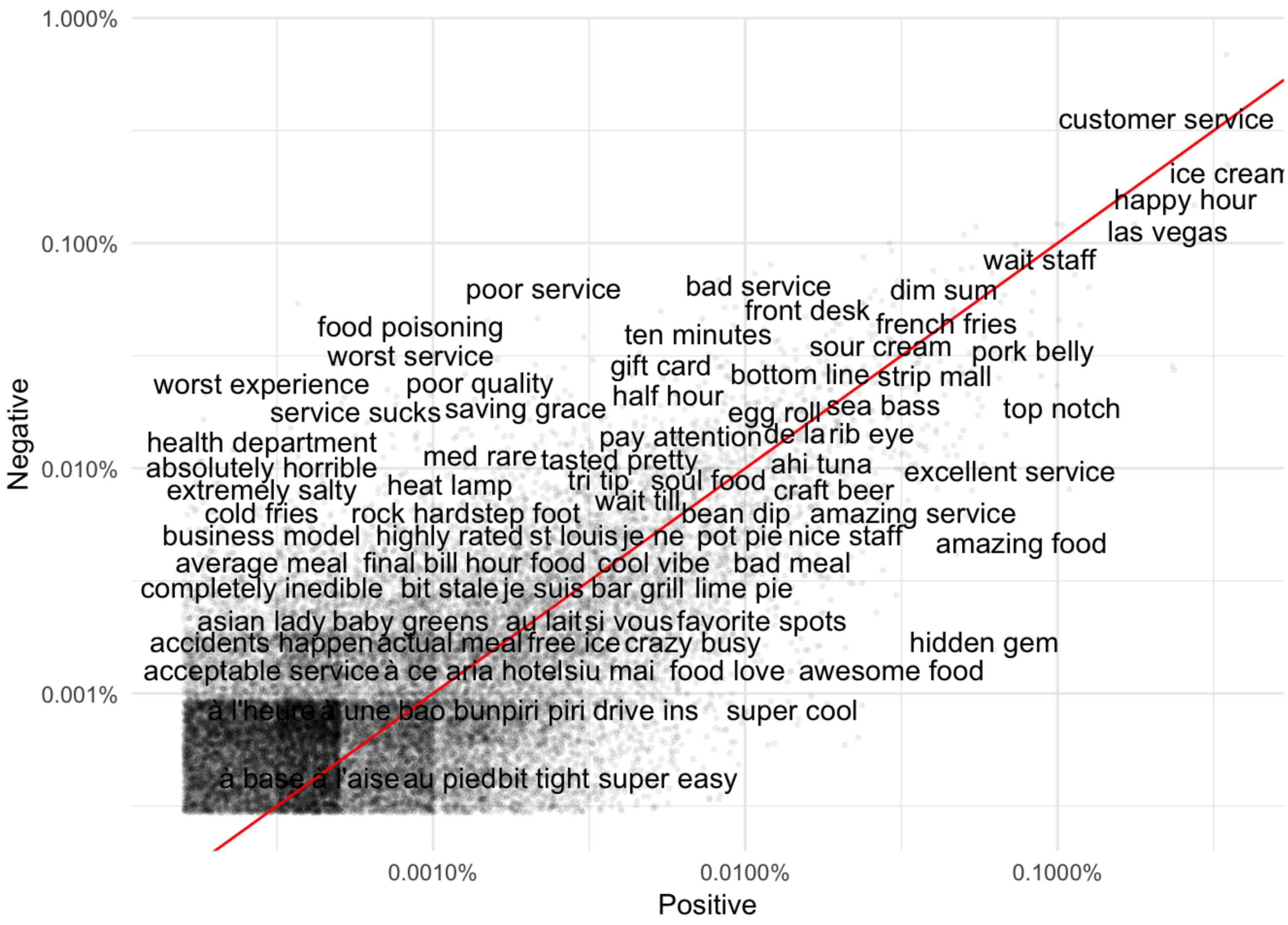
```
bigrams %>%
  select(bigram, group) %>%
  group_by(group) %>%
  count(bigram, group, sort = TRUE) %>%
  top_n(35, n) %>%
  ungroup() %>%
  ggplot(aes(reorder_within(bigram, n, group), n)) +
  geom_col(fill = "steelblue") +
  scale_x_reordered() +
  labs(x = "", y = "") +
  coord_flip() +
  theme_minimal() +
  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")
```

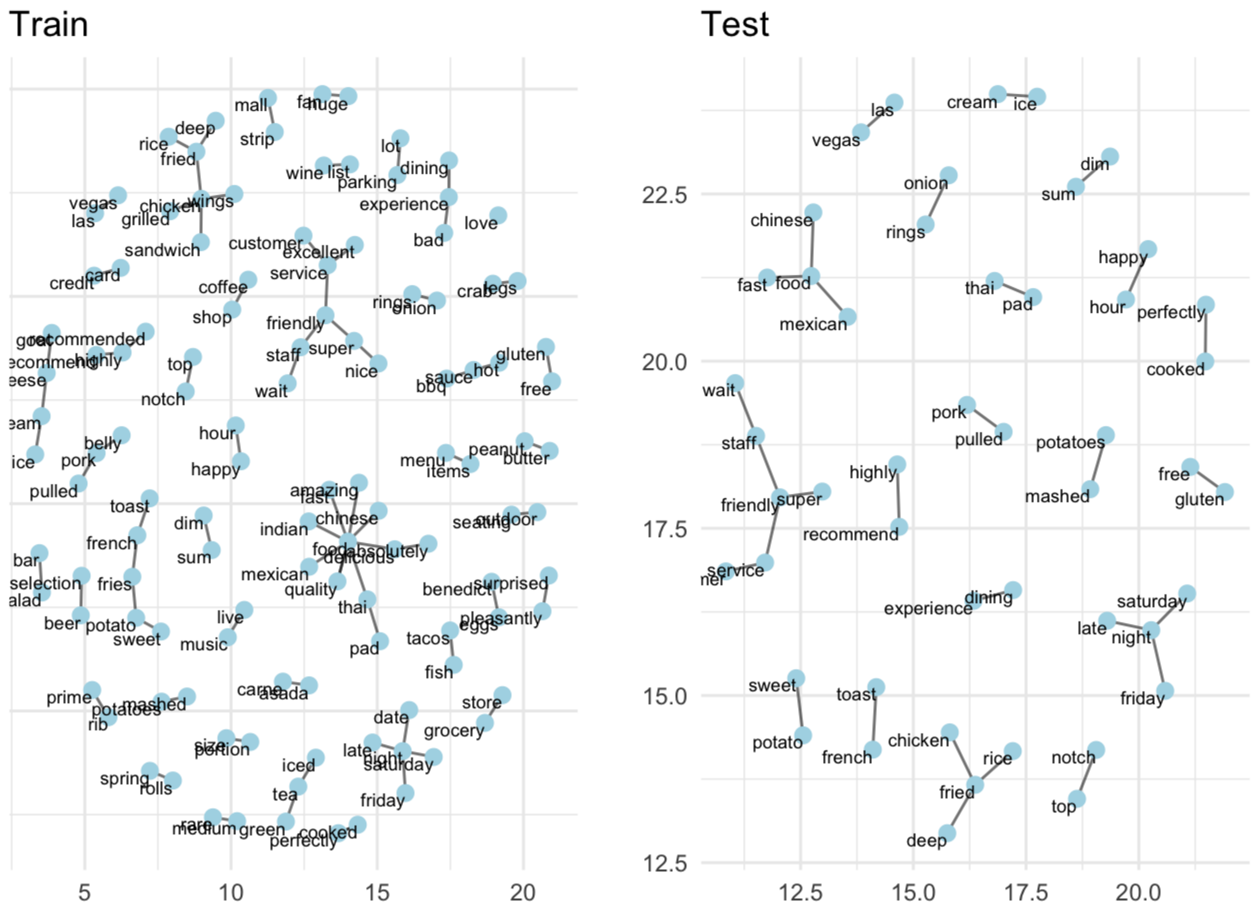


```
bigrams %>%
  group_by(Target) %>%
  count(bigram, sort = TRUE) %>%
  left_join(bigrams %>%
              group_by(Target) %>%
              summarise(total = n()), by = "Target") %>%
  mutate(freq = n/total) %>%
  select(Target, bigram, freq) %>%
  spread(Target, freq) %>%
  arrange(`Positive`, `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()
```



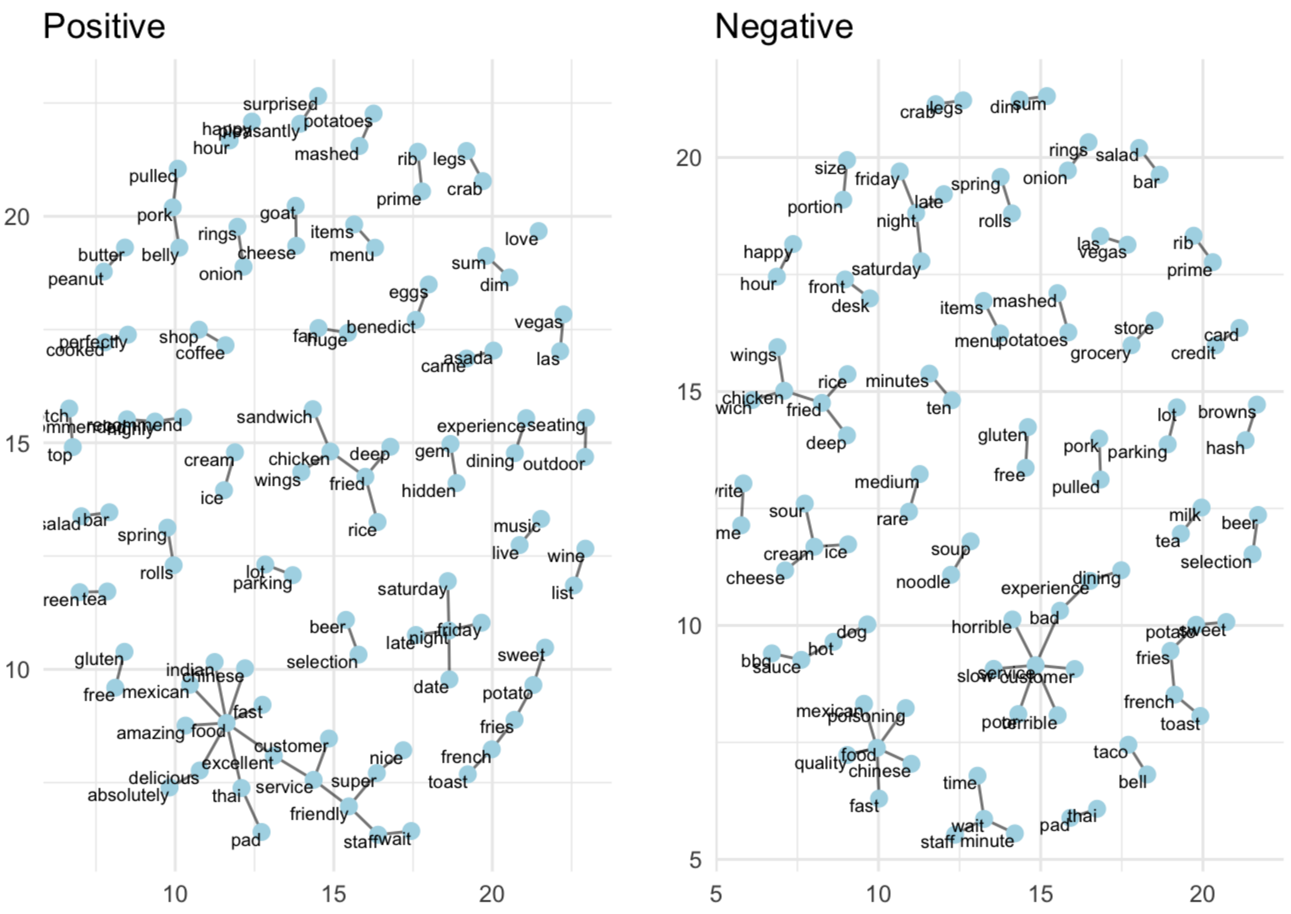
Network of Bigrams (Data)

```
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)
```



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 = "") +
  ggtitle("Negative") +
  theme_minimal()

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



Sentiment Analysis

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

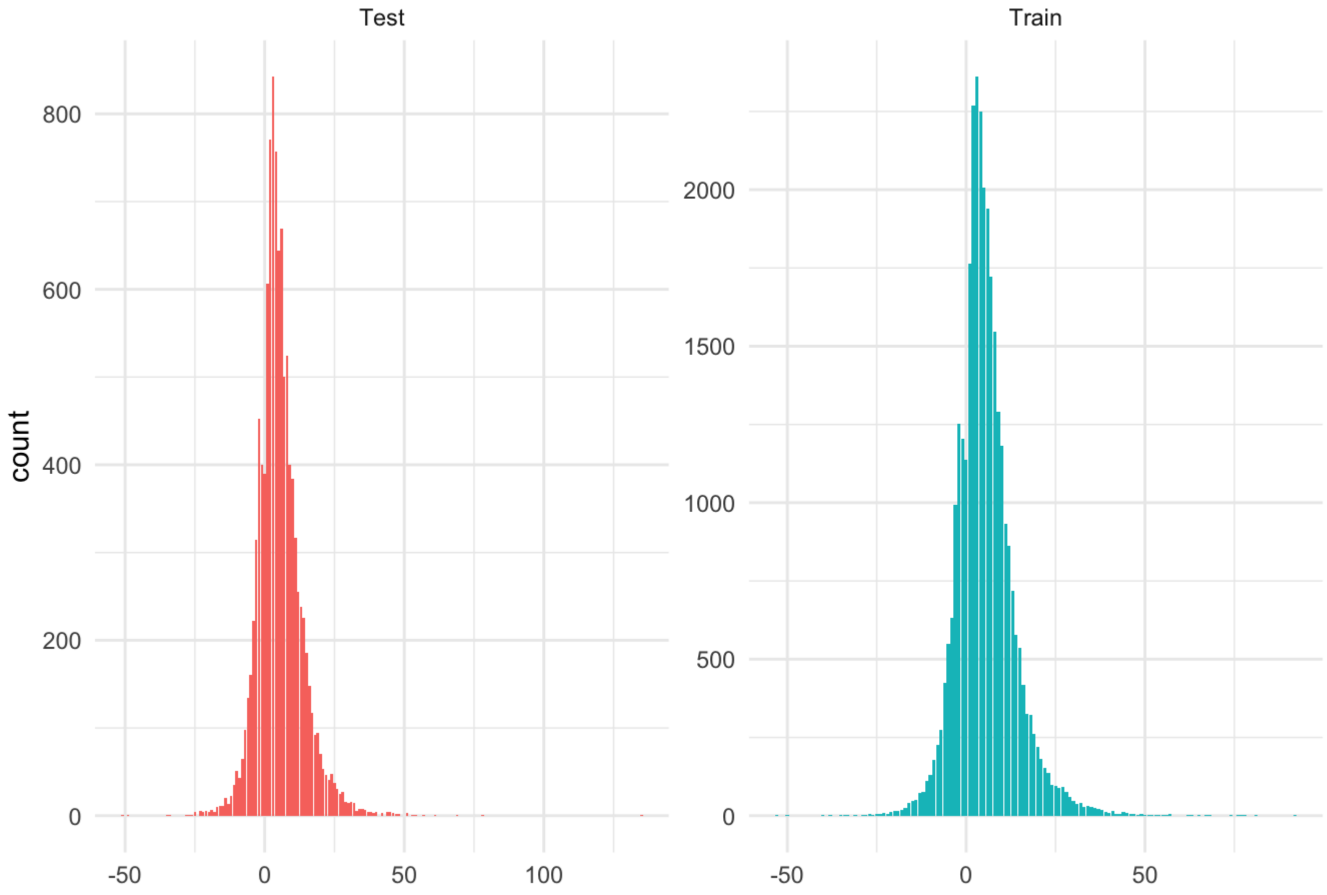
word	sentiment	lexicon	score
unfulfilled	negative	bing	NA
cessation	negative	nrc	NA
defy	negative	nrc	NA
deportation	sadness	nrc	NA
pertinacious	negative	bing	NA
fortune	positive	nrc	NA
perversion	negative	nrc	NA
doubt	sadness	nrc	NA
deadlocking	negative	loughran	NA
retort	negative	nrc	NA

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

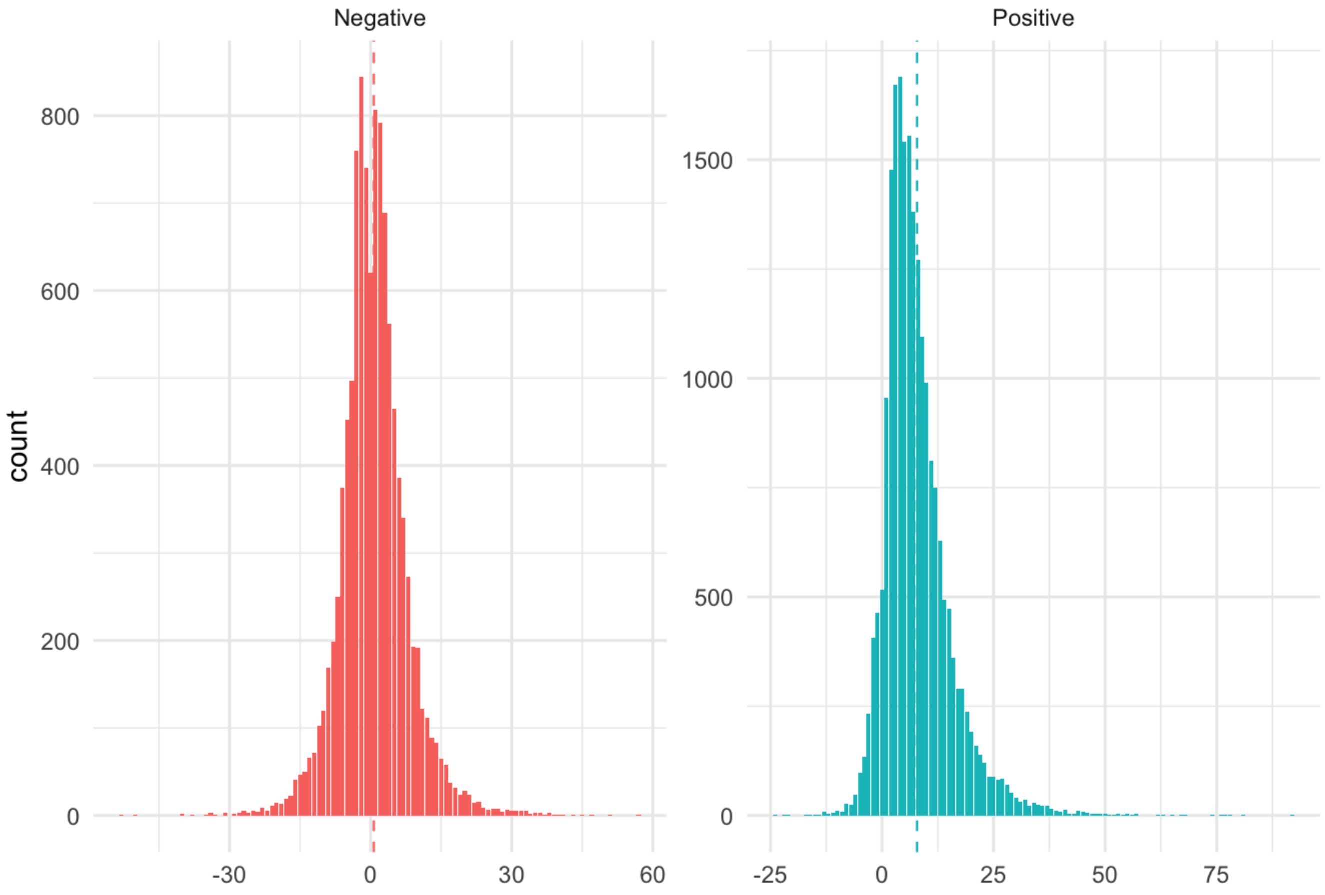
word	sentiment	lexicon	score
burdens	NA	AFINN	-2
bitches	NA	AFINN	-5
discontented	NA	AFINN	-2
praise	NA	AFINN	3
legally	NA	AFINN	1
stupidly	NA	AFINN	-2
demand	NA	AFINN	-1
favorited	NA	AFINN	2
waste	NA	AFINN	-1
misbehaves	NA	AFINN	-2

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()
```

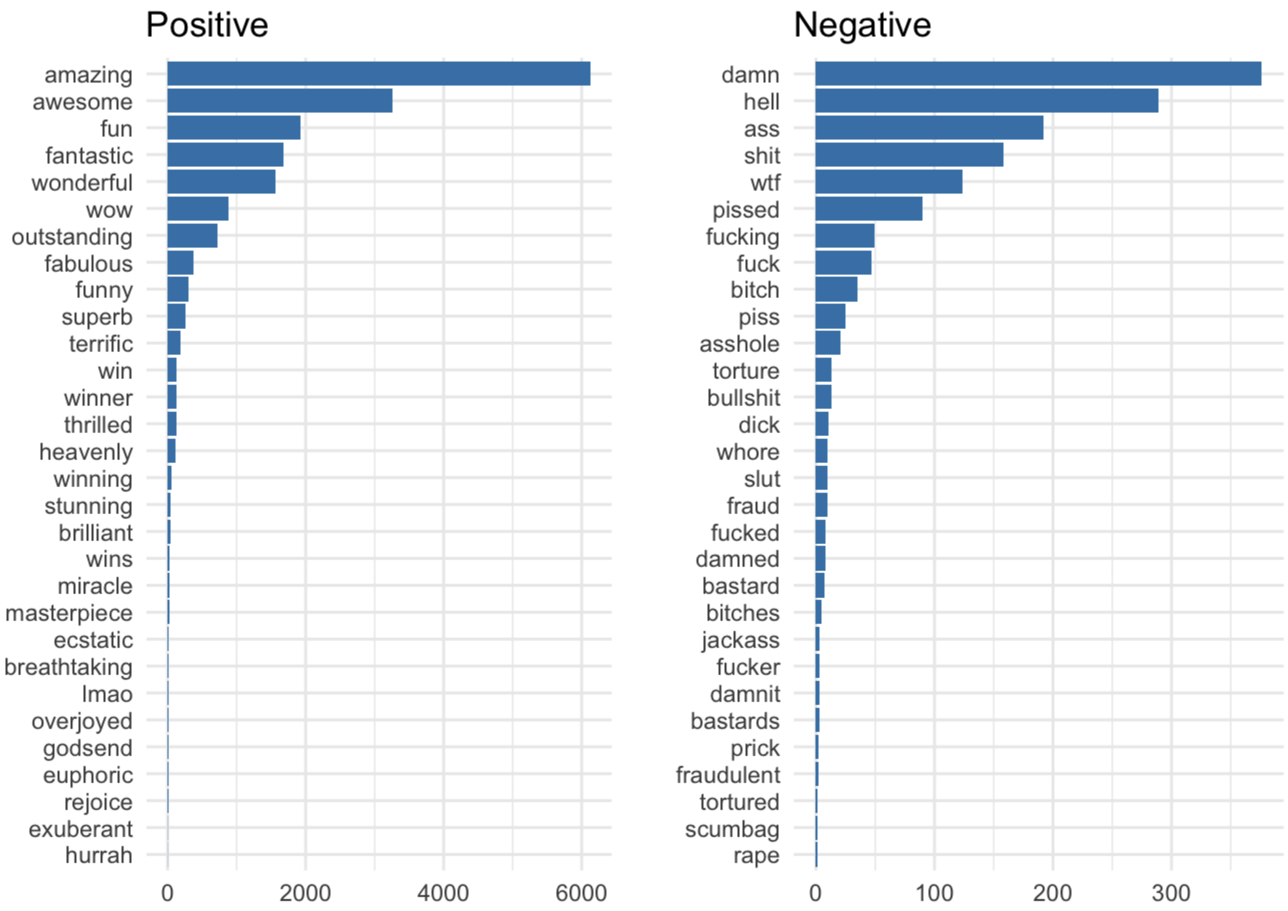


```
tokens_sent %>%
  filter(group == "Train") %>%
  group_by(review_id, Target) %>%
  summarise(score = sum(score)) %>%
  ungroup() %>%
  group_by(Target) %>%
  mutate(avg = mean(score)) %>%
  ungroup() %>%
  ggplot(aes(score, fill = factor(Target))) +
  geom_bar(show.legend = FALSE) +
  geom_vline(aes(xintercept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



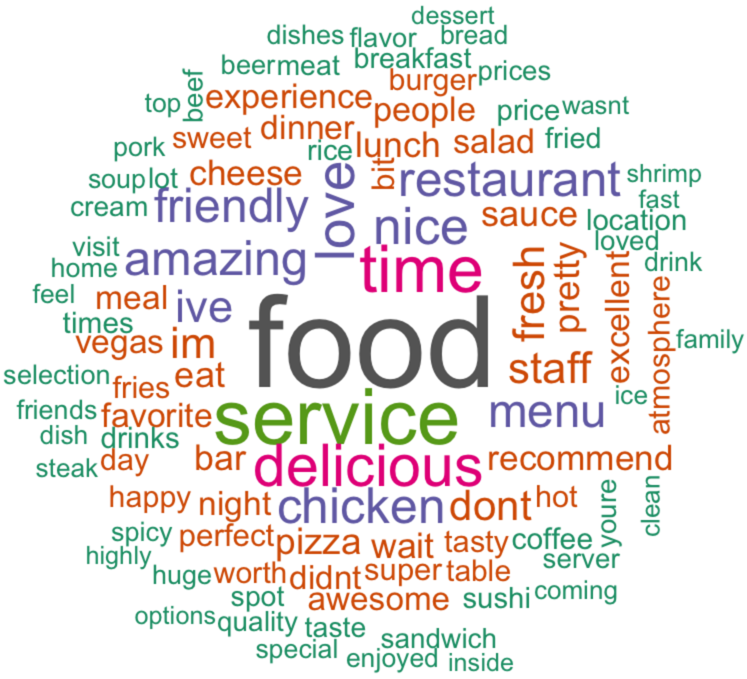
Most common words

```
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)
```

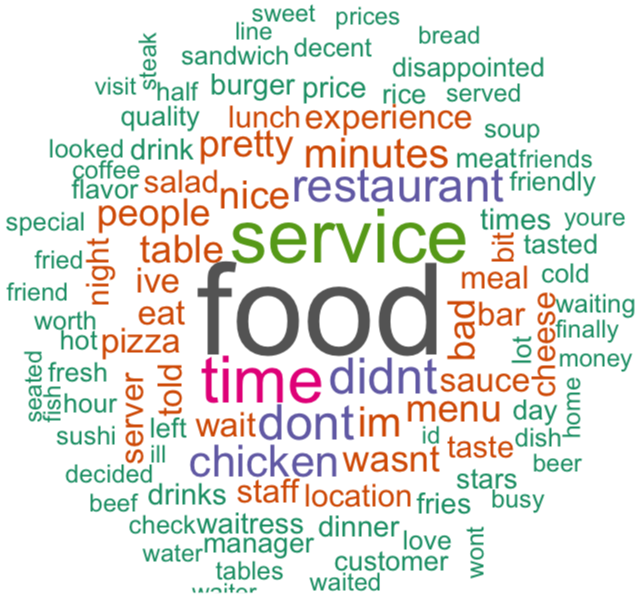


Wordcloud (Group)

```
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"))
```

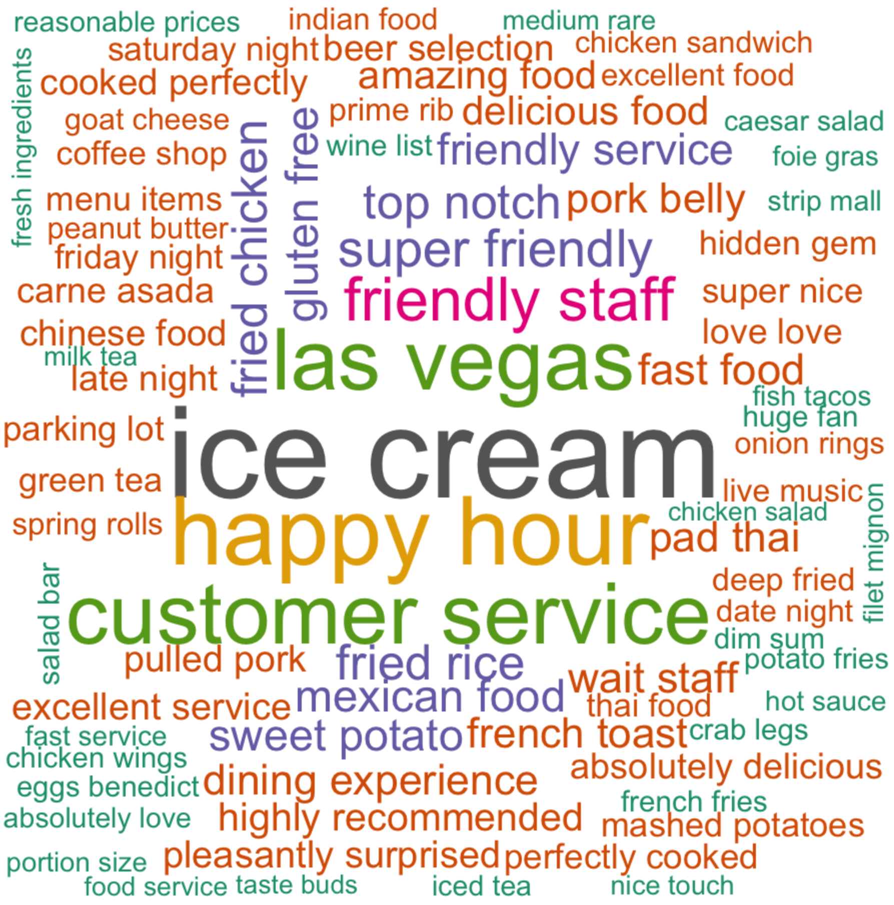


```
counts <- tokens %>%
  filter(Target == "Negative") %>%
  count(word, sort = TRUE) %>%
  top_n(100, n)
wordcloud(counts$word, counts$n, random.order = FALSE, colors =
RColorBrewer::brewer.pal(8,"Dark2"))
```

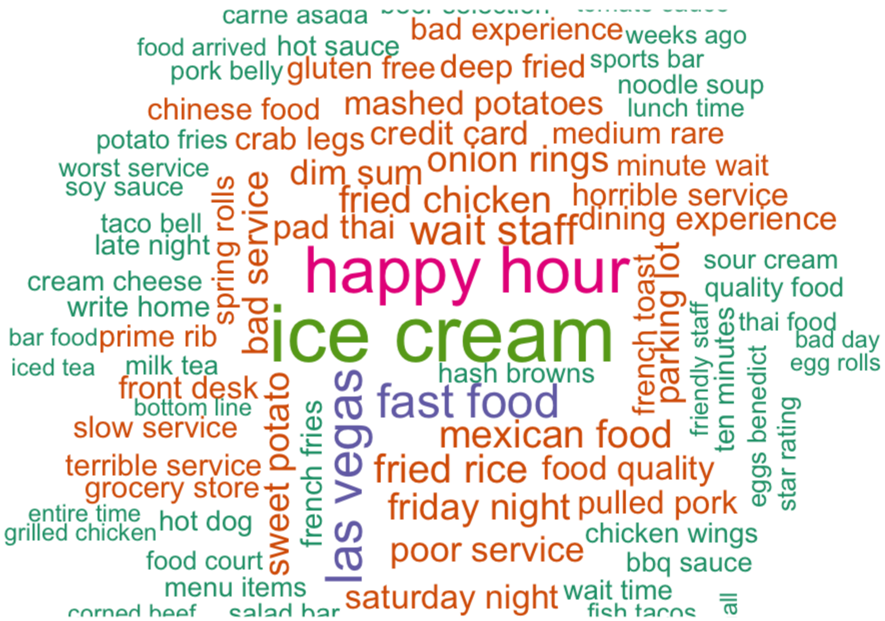


Wordcloud Bigram

```
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"))
```



```
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"))
```



Model Building

Remove Punctuations

```
puncts <- c(',', '.', '"', ':', ')', '(', '-', '!', '?', '|', ';',
"'", '$', '&', '/', '[', ']', '>', '%', '=', '#', '*', '+', '\\', '•',
'~', '@', '£', '·', '_', '{', '}', '©', '^', '®', '`', '<', '→', '°',
'€', '™', '›', '♥', '←', '×', '§', '″', '′', 'Â', '█', '½', 'à', '…',
'“', '★', '”', '–', '●', 'â', '►', '−', '¢', '²', '¬', '░', '¶', '↑',
'±', '¿', '▾', '═', '¦', '║', '―', '¥', '▓', '—', '‹', '─', '▒', '：',
'¼', '⊕', '▼', '▪', '†', '■', '’', '▀', '¨', '▄', '♫', '☆', 'é', '¯',
'♦', '¤', '▲', 'è', '¸', '¾', 'Ã', '⋅', '‘', '∞', '∙', '）', '↓', '、',
'│', '（', '»', '，', '♪', '╩', '╚', '³', '・', '╦', '╣', '╔', '╗',
'▬', '❤', 'ï', 'Ø', '¹', '≤', '‡', '√')
```

```
puncts <- paste(puncts, collapse = "|")
puncts <- paste("([", puncts, "])", sep = "", collapse = "")

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

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, ]
y_tr <- y[-val]
X_val <- sequences[train_index, ][val, ]
y_val <- y[val]
X_te <- sequences[-train_index, ]
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()
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"
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.