Sentiment Analysis and ABSA

Sirapob Lurojruang - 2215107

Contents


```
library(tokenizers)
library(tidyverse)
library(tidytext)
library(topicmodels)
library(textstem)
library(tm)
library(dplyr)
library(qdapRegex)
library(ggplot2)
library(ggrepel)
library(word2vec)
library(sentimentr)
```

```
library(lexicon)
library(SnowballC)
library(wordcloud)
library(RColorBrewer)
```

1 Overview of dataset

```
review <- read.csv(file = "Restaurant_Reviews.csv")
str(review)
```

```
## 'data.frame':    1000 obs. of  2 variables:
##  $ Review: chr  "Wow... Loved this place." "Crust is not good." "Not tasty and the texture was just n
##  $ Liked : int  1 0 0 1 1 0 0 0 1 1 ...
```

1.1 Text cleaning

Remove entry with no comment

```
review <- review%>%
  filter(Review != "")
```

Formatting

```
# Replace all html tags and non-alphanumeric characters
review$text1 <- str_remove_all(review$Review, "<.*?>") %>%
  str_replace_all("[^[:alnum:][:punct:]]", " ") %>%
   str_replace_all("\\s{2,}", " ")

# Remove all special characters, excluding apostrophe, using regex
review$text1 <- review$text1 %>%
  gsub("[^'[:alpha:][:space:]]", " ",.)

# remove extra spaces
review$text1 <- str_squish(review$text1)

# lowercase
review$text1 <- tolower(review$text1)
```

Create Index

```
# create review_id represent the unique identifier of each review
review <- review%>%
  mutate(review_id = row_number())
```

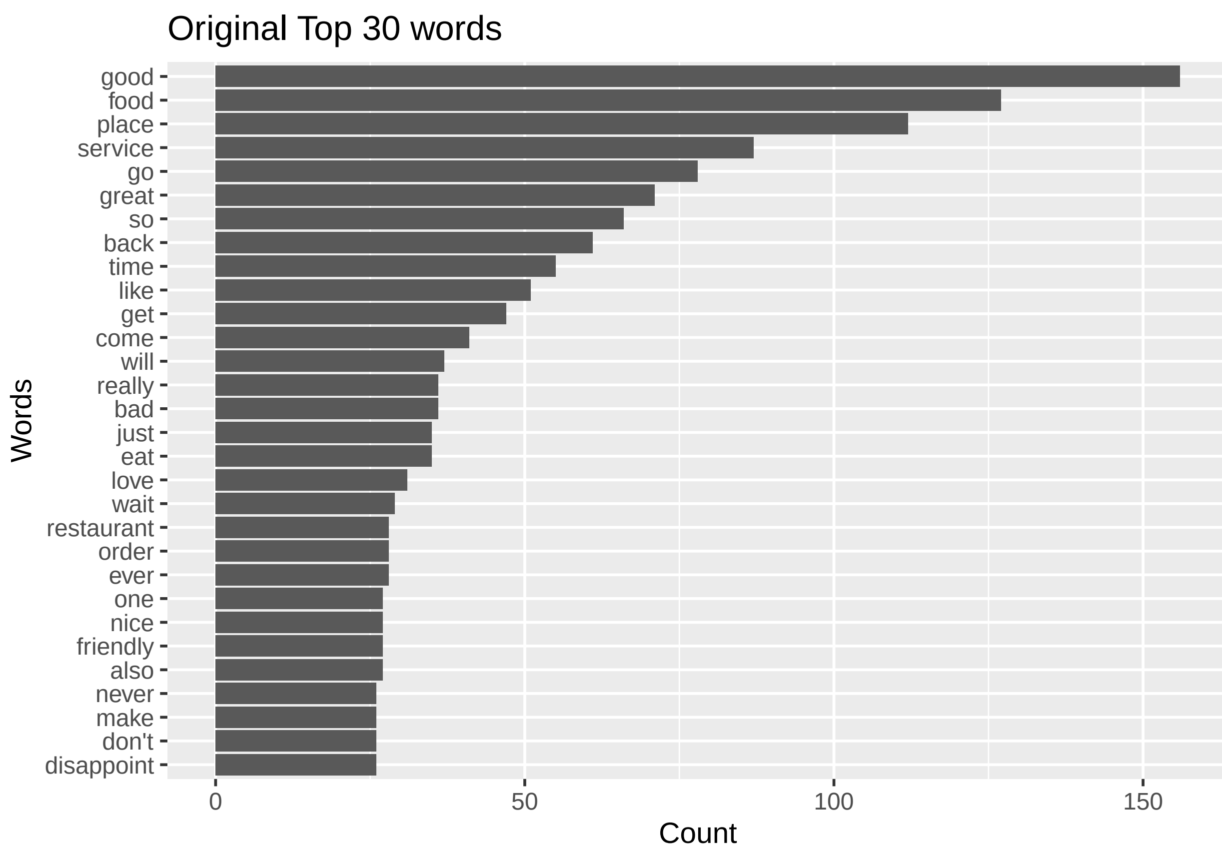
1.2 Tokenize, Removing stop words and Lemmatization

```
# Define a regular expression pattern to match hyphenated words
hyphen_pattern <- "[[:alnum:]]+(?:[-'][[:alnum:]]+)*"

# Define a custom tokenization function using the hyphen_pattern
custom_tokenize <- function(x) {
  str_extract_all(x, hyphen_pattern)
}

# Decide to use stop words that only in 2 out of 3 lexicons
stop_words_use <- stop_words%>%
  group_by(word)%>%
  summarise(count = n())%>%
  filter(count > 2)

# Remove stop words, lemmetized and save data in token level
token <- review%>%
  unnest_tokens(text1,output=word_token,token=custom_tokenize)%>%
  anti_join(stop_words_use,by=c("word_token"="word"))%>%
  mutate(word_lemma = lemmatize_words(word_token)) %>%
  unnest(word_lemma)
```

1.3 Exploring word token

Visualise what is the top 30 words

```
token%>%
  group_by(word_lemma) %>%
  count() %>%
  arrange(desc(n))%>%
  head(30) %>%
  ggplot(.,aes(y=reorder(word_lemma,n),x=n))+geom_bar(stat='identity')+labs(x = "Count", y = "Words", ti
```



2 Topic modelling

2.1 Calculate TFIDF

Calculate count and TFIDF for each term

```
y_counts <- token %>%
  count(review_id, word_lemma, sort = TRUE) %>%
  ungroup() %>%
  rename(count=n) %>%
  arrange(desc(count))

y_tfidf <- y_counts %>%
bind_tf_idf(review_id, word_lemma, count)

head(y_tfidf)
```

```
## # A tibble: 6 x 6
##   review_id word_lemma count    tf   idf tf_idf
##       <int> <chr>      <int> <dbl> <dbl>  <dbl>
## 1       124 steak          4 0.222  4.99  1.11
## 2       237 sauce          3 0.273  4.90  1.34
## 3       237 say            3 0.115  4.90  0.565
```

```
## 4       453 food           3 0.0236  4.90  0.116
## 5       538 great          3 0.0423  5.59  0.236
## 6       573 wait           3 0.103   7.39  0.764
```

Explore what is top words for each review

```
# Group the y_tfidf dataframe by review_id
y_tfidf_grouped <- y_tfidf %>%
  group_by(review_id) %>%
  # Arrange the rows within each group by decreasing tf_idf values
  arrange(desc(tf_idf)) %>%
  # Select the top 5 rows within each group
  slice_head(n = 5) %>%
  # Unnest the word_token column
  unnest(word_lemma) %>%
  # Create a row number variable within each group
  group_by(review_id) %>%
  mutate(row_num = row_number()) %>%
  # Spread the top 5 words into separate columns, with row_num as the ID variable
  pivot_wider(id_cols = review_id, names_from = row_num, values_from = word_lemma, names_prefix = "word

# Select only the review_id and word columns
top_5_words <- y_tfidf_grouped %>%
  select(review_id, starts_with("word_"))

# Rename the columns to remove the "word_" prefix
colnames(top_5_words) <- paste0("top_", 0:5)
colnames(top_5_words)[1] <- "review_id"
# Print the resulting dataframe
head(top_5_words)
```

```
## # A tibble: 6 x 6
## # Groups:   review_id [6]
##   review_id top_1     top_2   top_3 top_4 top_5
##       <int> <chr>     <chr>   <chr> <chr> <chr>
## 1         1 wow       love    place <NA>  <NA>
## 2         2 crust     good    <NA>  <NA>  <NA>
## 3         3 nasty     texture tasty just  <NA>
## 4         4 bank      holiday rick  steve recommendation
## 5         5 selection menu    price so    great
## 6         6 angry     damn    be    pho   now
```

2.2 Create DTM

Create count DTM

```
# Create DTM
dtm_count <- y_tfidf %>%
  cast_dtm(review_id,word_lemma,count)
dtm_count
```

```
## <<DocumentTermMatrix (documents: 1000, terms: 1612)>>
```

```
## Non-/sparse entries: 5649/1606351
## Sparsity           : 100%
## Maximal term length: 17
## Weighting          : term frequency (tf)
```

```
# Store DTM with sparse term removed separately (For further use)
dtm_count_sp <- removeSparseTerms(dtm_count,0.99)
dtm_count_sp
```

```
## <<DocumentTermMatrix (documents: 1000, terms: 95)>>
## Non-/sparse entries: 2452/92548
## Sparsity           : 97%
## Maximal term length: 10
## Weighting          : term frequency (tf)
```

```
as.matrix((dtm_count[1:5,1:5]))
```

```
##      Terms
## Docs  steak sauce say food great
##   124     4     0   1    0     0
##   237     0     3   3    0     0
##   453     0     0   0    3     0
##   538     1     0   0    0     3
##   573     0     0   0    0     0
```

2.3 LDA and Topics

2.3.1 Set-up LDA model

```
# Use LDA to calcualte Beta
set.seed(12345)
my_topic_model <- LDA(dtm_count,k = 4,method = "Gibbs")
topics <- tidy(my_topic_model, matrix = "beta")
```

2.3.2 Calculate perplexity

```
perplexity(my_topic_model, newdata = dtm_count)
```

```
## [1] 642.3591
```

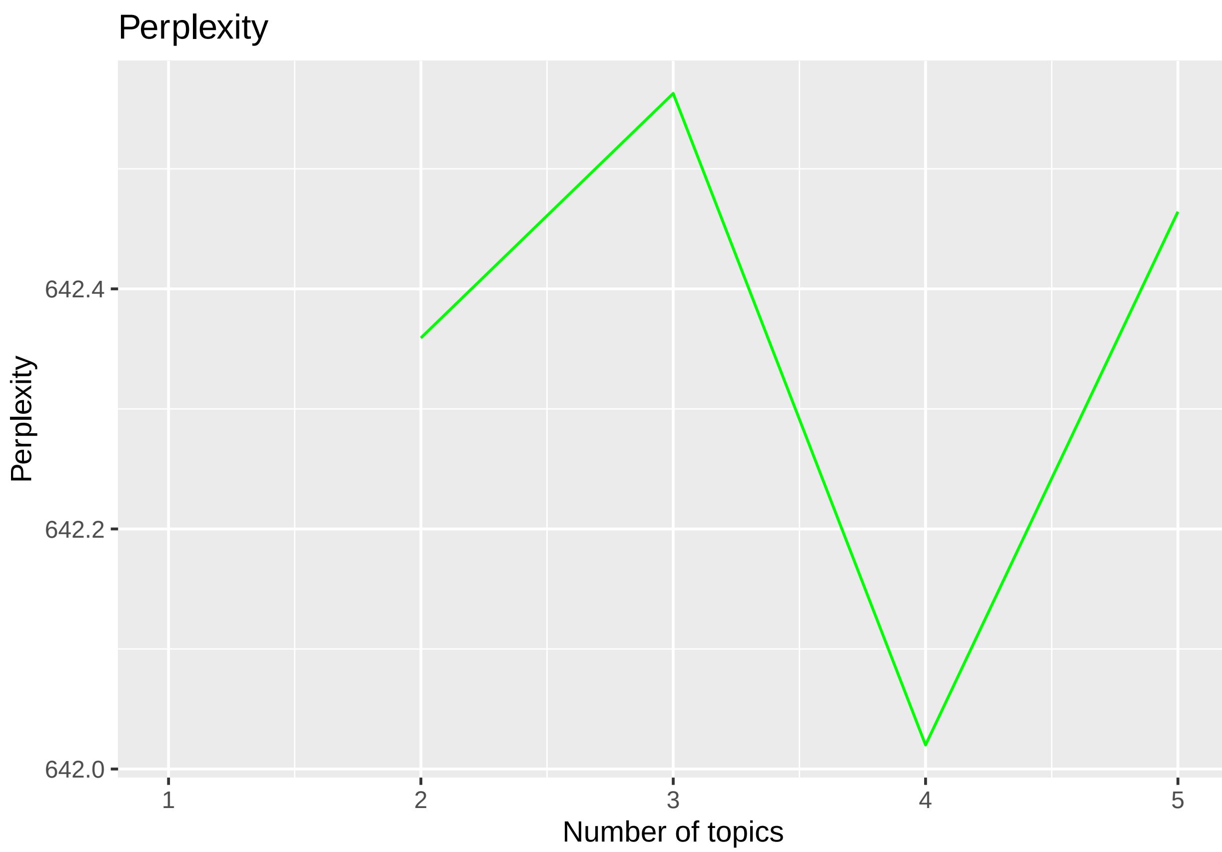
Finding the minimum perplexity value from 2 to 5 topics

```
set.seed(12345)
k_topics <- c(2:5)
perplexity_df <- data.frame(perp_value=numeric())
for (i in k_topics){
  fitted <- LDA(dtm_count, k = i, method = "Gibbs")
  perplexity_df[i,1] <- perplexity(my_topic_model,dtm_count)
}
```

Plot the results « 4 is the best

```
g <- ggplot(data=perplexity_df, aes(x= as.numeric(row.names(perplexity_df)))) + labs(y="Perplexity",x="]

g <- g + geom_line(aes(y=perp_value), colour="green")
g
```

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```



2.3.3 Inspect Topics and Probabilities

Inspect the topics and their probabilities for each document using the posterior function from the topicmodels package.

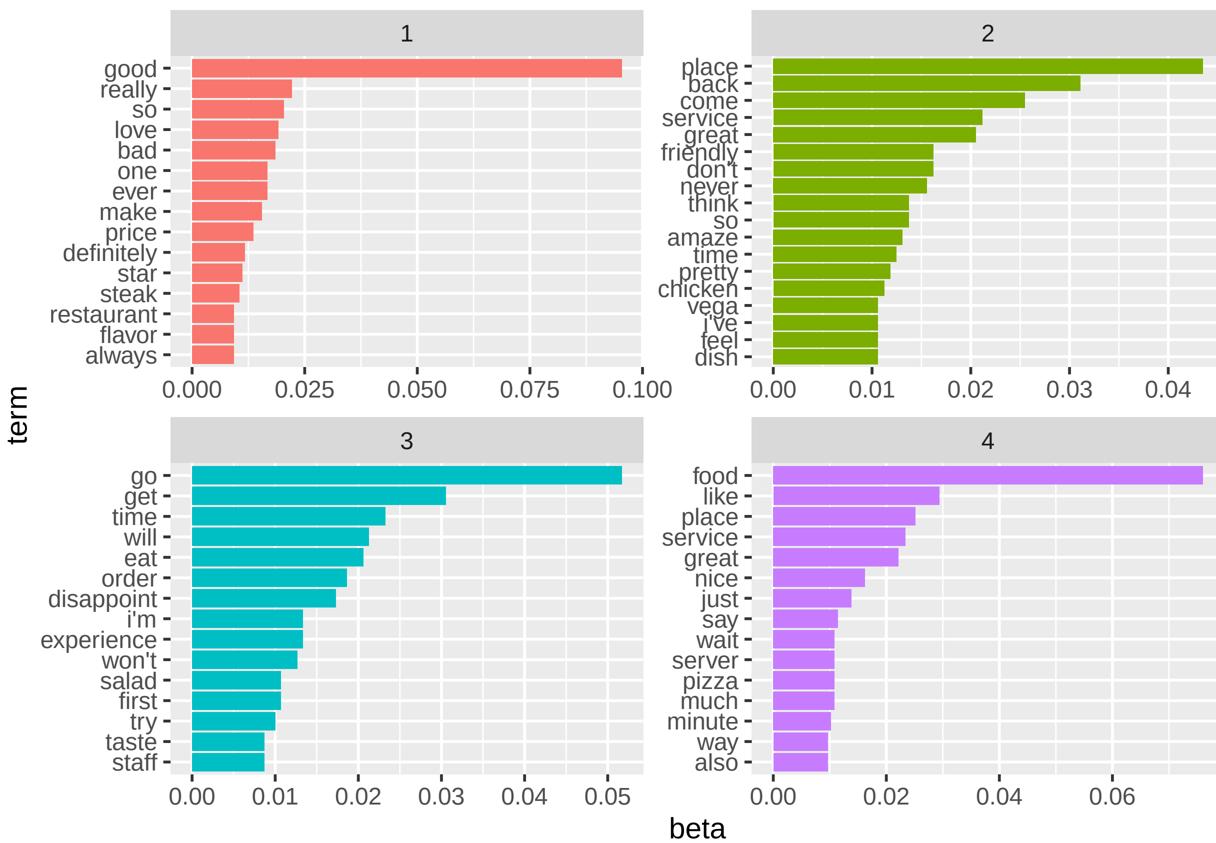
```
topics_prob <- posterior(my_topic_model)$topics
terms <- terms(my_topic_model, 5)
terms_all <- terms(my_topic_model)
colnames(topics_prob) <- apply(terms, 2, paste, collapse = ",")
head(topics_prob)
```

```
##     good,really,so,love,bad place,back,come,service,great go,get,time,will,eat
## 124               0.3307692                     0.2384615          0.2230769
## 237               0.2279412                     0.1985294          0.2720588
## 453               0.2692308                     0.2384615          0.2230769
```

```
## 538              0.2500000                         0.2672414            0.2327586
## 573              0.2358491                         0.2547170            0.2735849
## 624              0.2500000                         0.2045455            0.2348485
##     food,like,place,service,great
## 124                     0.2076923
## 237                     0.3014706
## 453                     0.2692308
## 538                     0.2500000
## 573                     0.2358491
## 624                     0.3106061
```

Plot chart based on beta of term on each topics

```
top_terms <- topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 15) %>%
  ungroup() %>%
  arrange(topic, desc(beta))

top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~ topic, scales = "free") +
    scale_y_reordered()
```

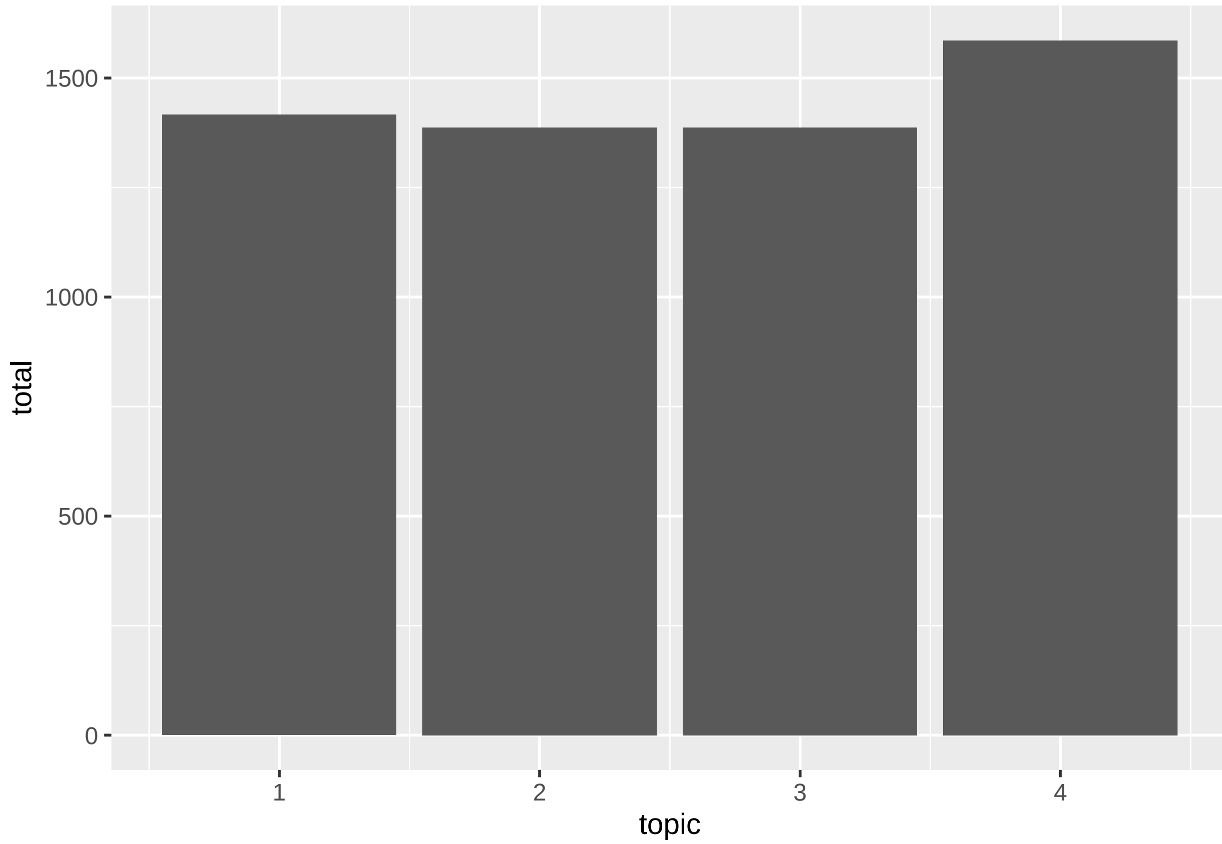


2.3.4 Assign each term to a topic

```
#Assign each term to each topics
assignments <- augment(my_topic_model, data = dtm_count)

# Store mapping between term and topics
map_top_terms <- assignments%>%
  group_by(term)%>%
  summarise(topic = max(.topic), freq = sum(count))

# Check top terms based on freq
assignments%>%
  group_by(term)%>%
  summarise(topic = max(.topic), freq = sum(count))%>%
  group_by(topic)%>%
  slice_max(freq, n = 5)%>%
  arrange(topic,desc(freq))
```

```
## # A tibble: 21 x 3
## # Groups:   topic [4]
##    term     topic  freq
##    <chr>    <dbl> <dbl>
##  1 good         1   156
##  2 so           1    66
##  3 bad          1    36
##  4 really       1    36
##  5 love         1    31
##  6 place        2   112
##  7 back         2    61
##  8 come         2    41
##  9 friendly     2    27
## 10 don't        2    26
## # i 11 more rows
```

```
# Check distribution of the topics
map_top_terms%>%
  group_by(topic)%>%
  summarise(total = sum(freq))%>%
  ggplot()+
    geom_col(mapping = aes(x = topic, y = total))
```



Get the top terms for each topics

2.3.5 Name the Topics

Extract top terms of each topics to help the naming

```
k = max(top_terms$topic)
for (n in 1:k){
topic_name <- paste0("t",n)
result <- assign(topic_name,top_terms%>%
  filter(topic == n)%>%
  select(term))

my_string <- paste0(result$term,collapse = ",")

print(my_string)
}
```

```
## [1] "good,really,so,love,bad,one,ever,make,price,definitely,star,steak,flavor,restaurant,always"
## [1] "place,back,come,service,great,friendly,don't,never,so,think,amaze,time,pretty,chicken,i've,feel
## [1] "go,get,time,will,eat,order,disappoint,experience,i'm,won't,salad,first,try,taste,staff"
## [1] "food,like,place,service,great,nice,just,say,wait,pizza,server,much,minute,also,way"
```

According to Chat GPT Topic 1: Overall Dining Experience Topic 2: Atmosphere and Service Topic 3: Dining Out Experience Topic 4: Food and Service Quality

Create word clouds to help name the topics based on frequency

```
# Word clouds
set.seed(1234)

for (k in 1:4){
wc <- map_top_terms%>%filter(topic == k)
wordcloud(words = wc$term, freq = wc$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
}
```


place
back
come
friendly
don't
never
think
vega
chicken
amaze
dish
take
feel
i've
pretty
menu
can
selection
lunch
slow


go
time
get
will
eat
disappoint
experience
order
taste
i'm
staff
first
salad
fry
won't
atmosphere
breakfast
dine
try
bland
fantastic
everything
sauce
ask
terrible
waitress
worth
find

Final naming of aspect

```
# Name each aspect based on GPT and Word clouds
map_top_terms <- map_top_terms%>%
  mutate(aspect = case_when(
    topic == 1 ~ "Value-to-money",
    topic == 2 ~ "Atmosphere",
    topic == 3 ~ "Experience",
    topic == 4 ~ "Food and Service"
  ))
```

Final topics selection 1. Value-to-money 2. Atmosphere 3. Experience 4. Food and Service

3 Sentiment Analysis

3.1 Lexicon

Download bing lexicon

3.2 Assign sentiment value to token

```r
# Attached Sentiment to tokens
token_sen <- token%>%
  left_join( bing_dictionary_v, by = c("word_lemma" = "word"))

# Attached Sentiment and topics to tokens
token_top_sen <- token%>%
  left_join(map_top_terms, by = c("word_lemma" = "term"))%>%
  left_join(bing_dictionary_v, by = c("word_lemma" = "word"))

# Sentiment Analysis as input
df_sen <- token_sen%>%
  group_by(review_id,Liked)%>%
  summarise(sentiment = round(mean(sentiment, na.rm=TRUE),3))%>%
  dplyr::mutate(sentiment = replace_na(sentiment,0))
```

```
## `summarise()` has grouped output by 'review_id'. You can override using the
## `.groups` argument.
```

```r
# ABSA as input
df_top_sen <- token_top_sen%>%
  group_by(review_id,aspect,Liked)%>%
  summarise(sentiment = round(sum(sentiment, na.rm=TRUE),3))%>%
  dplyr::mutate(sentiment = replace_na(sentiment,0))%>%
  pivot_wider(names_from = aspect, values_from = sentiment)
```

```
## `summarise()` has grouped output by 'review_id', 'aspect'. You can override
## using the `.groups` argument.
```

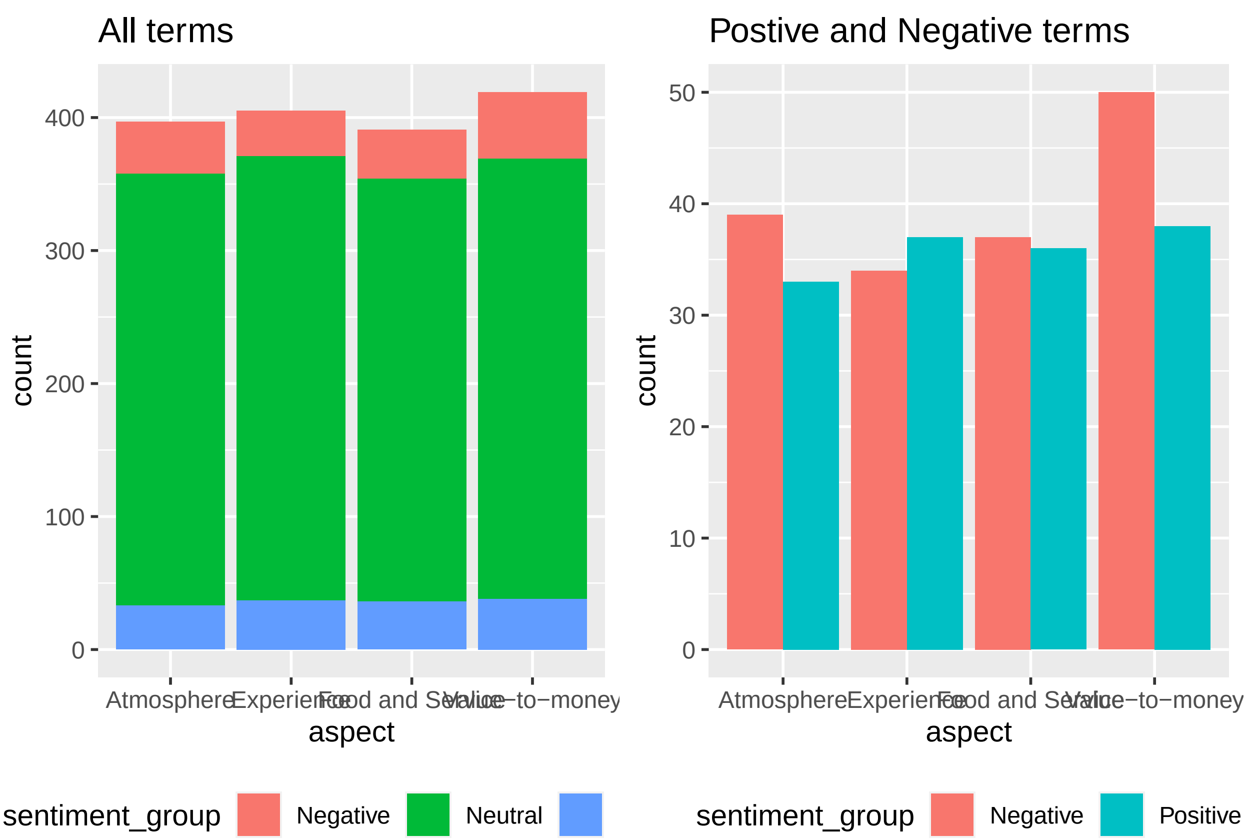
```r
0 -> df_top_sen[is.na(df_top_sen)] # Impute NA with 0
```

3.3 Check sentiment on each aspect

```r
gridExtra::grid.arrange(
token_top_sen%>%
  mutate(sentiment_group = case_when(sentiment == 1 ~ "Positive",
                                     sentiment == -1 ~ "Negative",
                                     sentiment == 0 ~ "Negative",
                                     is.na(sentiment) ~ "Neutral"))%>%
  group_by(aspect,sentiment_group)%>%
  summarise(count = n_distinct(word_lemma))%>%
  ggplot(aes(x = aspect,y = count, fill = sentiment_group))+
    geom_col(stat = "identity")+
    labs(title = "All terms")+
    theme(legend.position = "bottom"),
token_top_sen%>%
  mutate(sentiment_group = case_when(sentiment == 1 ~ "Positive",
                                     sentiment == -1 ~ "Negative",
                                     sentiment == 0 ~ "Negative",
                                     is.na(sentiment) ~ "Neutral"))%>%
```

```
  group_by(aspect,sentiment_group)%>%
  summarise(count = n_distinct(word_lemma))%>%
  filter(sentiment_group != "Neutral")%>%
  ggplot(aes(x = aspect,y = count, fill = sentiment_group))+
    geom_col(stat = "identity",position = position_dodge())+
    labs(title = "Postive and Negative terms")+
    theme(legend.position = "bottom")
, ncol =2)
```

```
## `summarise()` has grouped output by 'aspect'. You can override using the
## `.groups` argument.

## Warning in geom_col(stat = "identity"): Ignoring unknown parameters: `stat`

## `summarise()` has grouped output by 'aspect'. You can override using the
## `.groups` argument.

## Warning in geom_col(stat = "identity", position = position_dodge()): Ignoring
## unknown parameters: `stat`
```



- Above chart shows that "Neutral" words dominates all 4 aspects.
- Overall, there are many terms expressing negative sentiment for atmosphere and value where people are more postive towards experience. People having mixed sentiment towards Food & Service

4 Preditive model

4.1 Define predictive model

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
# SVM radial model
svm_rad_model <- function (data_train)
  {
  set.seed(12345)
  ctrl <- trainControl(method="repeatedcv",    # 10fold cross validation
                     repeats=5,          # do 5 repetitions of cv
                     summaryFunction=twoClassSummary,    # Use AUC to pick the best model
                     classProbs=TRUE,
                     savePredictions = T)
  return(caret::train(Liked~.,
                    data = data_train,
                    method = "svmRadial",tuneLength = 5, # 5 values of the cost function
                    preProc = c("center","scale"),  # Center and scale data
                    metric="ROC",
                   trControl=ctrl))
  }
```

4.2 Predicting

Testing out different models

```
# Turn input into Matrix
data_sen <- as.data.frame(as.matrix(df_sen))%>%ungroup()%>%select(-review_id)
data_top_sen <- as.data.frame(as.matrix(df_top_sen))%>%ungroup()%>%select(-review_id)

# Split the dataset into training and testing sets
set.seed(12345)
idx <- sample(1:nrow(data_sen), size = 0.8 * nrow(data_sen), replace = FALSE)

  data_sen$Liked <- as.factor(ifelse(data_sen$Liked == "1","Like","Not_like"))
  data_sen_train<-data_sen[idx,]
  data_sen_test<-data_sen[-idx,]

  data_top_sen$Liked <- as.factor(ifelse(data_top_sen$Liked == "1","Like","Not_like"))
  data_top_sen_train<-data_top_sen[idx,]
```

```r
  data_top_sen_test<-data_top_sen[-idx,]

# Build models for each input
Base_SVM <- svm_rad_model(data_sen_train)
ABSA_SVM <- svm_rad_model(data_top_sen_train)

# Store confusion matrix
Base_SVM_con <- confusionMatrix(predict(Base_SVM, data_sen_test),data_sen_test$Liked,
                                mode = "everything",
                                positive = "Like")
ABSA_SVM_con <- confusionMatrix(predict(ABSA_SVM, data_top_sen_test),data_top_sen_test$Liked,
                                mode = "everything",
                                positive = "Like")

# Check result
Base_SVM_con
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Like Not_like
##   Like       72       24
##   Not_like   25       79
##
##                Accuracy : 0.755
##                  95% CI : (0.6894, 0.8129)
##     No Information Rate : 0.515
##     P-Value [Acc > NIR] : 2.755e-12
##
##                   Kappa : 0.5094
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7423
##             Specificity : 0.7670
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.7596
##               Precision : 0.7500
##                  Recall : 0.7423
##                      F1 : 0.7461
##              Prevalence : 0.4850
##          Detection Rate : 0.3600
##    Detection Prevalence : 0.4800
##       Balanced Accuracy : 0.7546
##
##        'Positive' Class : Like
##
```

```r
ABSA_SVM_con
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

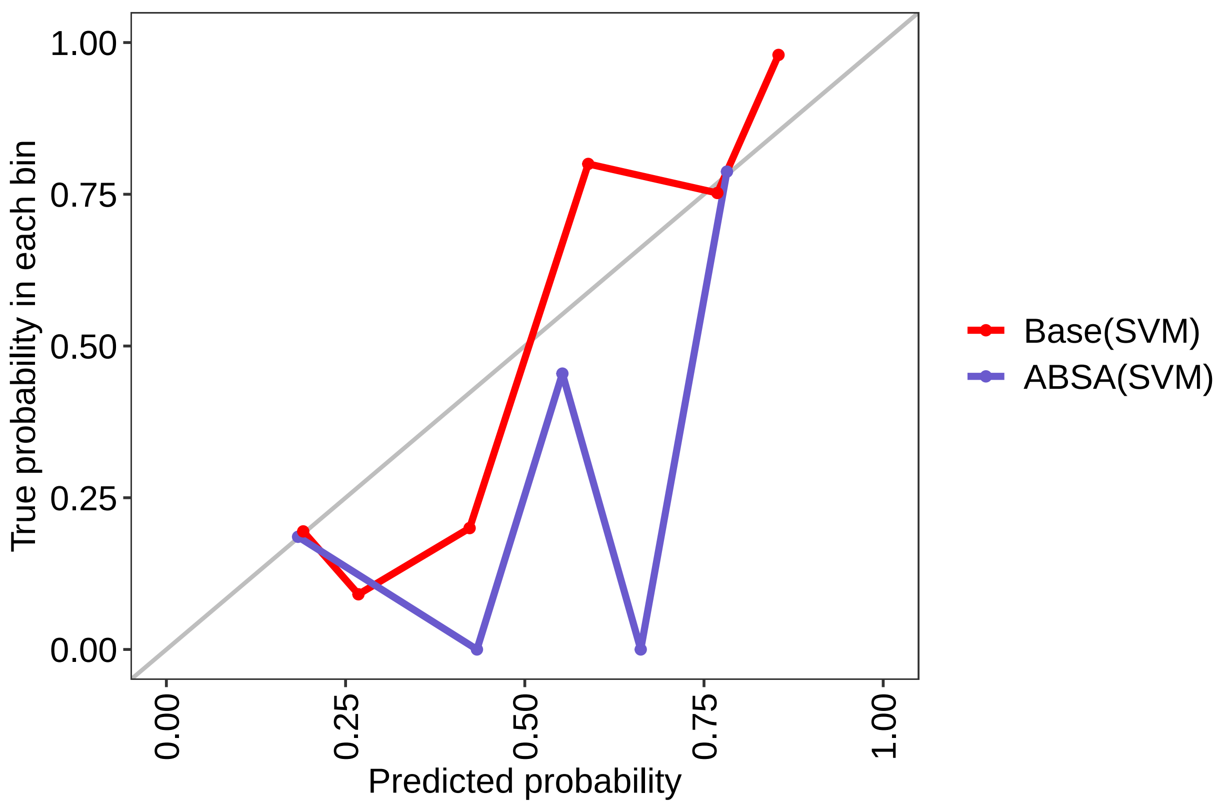
```
## Prediction Like Not_like
##   Like       75       26
##   Not_like   22       77
##
##                Accuracy : 0.76
##                  95% CI : (0.6947, 0.8174)
##     No Information Rate : 0.515
##     P-Value [Acc > NIR] : 9.305e-13
##
##                   Kappa : 0.5201
##
##  Mcnemar's Test P-Value : 0.665
##
##             Sensitivity : 0.7732
##             Specificity : 0.7476
##          Pos Pred Value : 0.7426
##          Neg Pred Value : 0.7778
##               Precision : 0.7426
##                  Recall : 0.7732
##                      F1 : 0.7576
##              Prevalence : 0.4850
##          Detection Rate : 0.3750
##    Detection Prevalence : 0.5050
##       Balanced Accuracy : 0.7604
##
##        'Positive' Class : Like
##
```

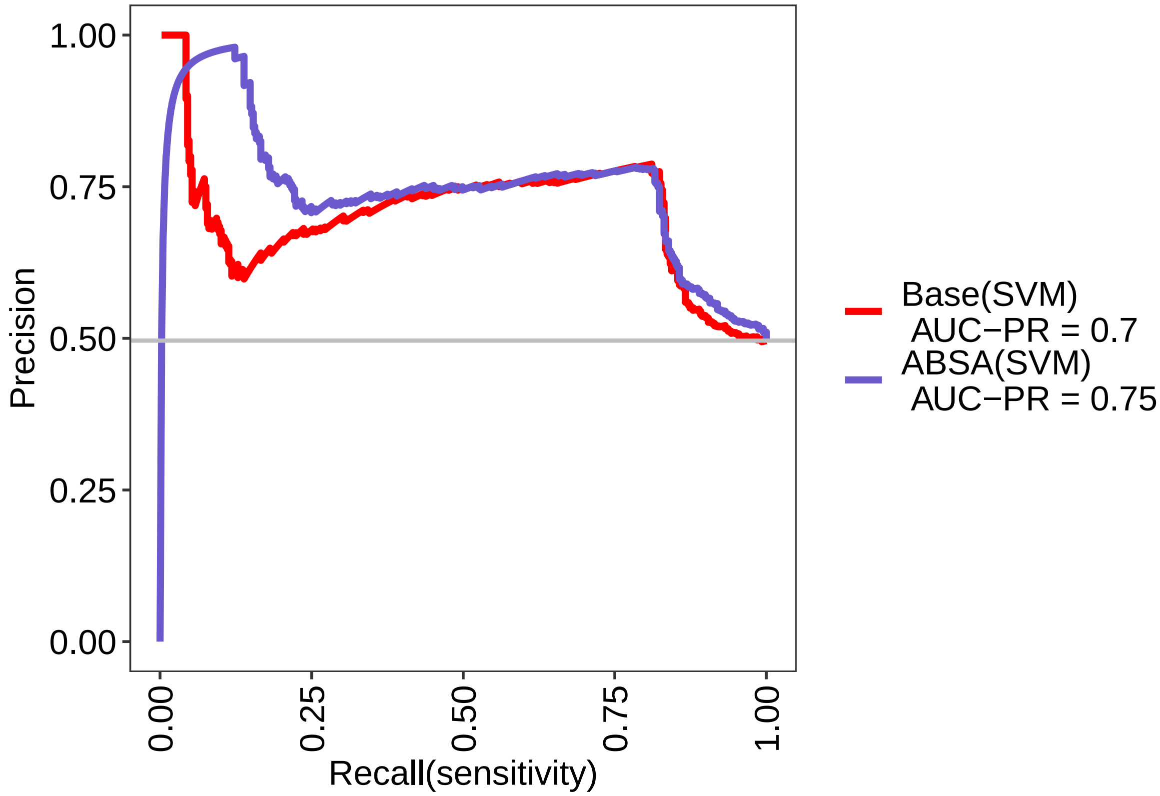
Plotting charts

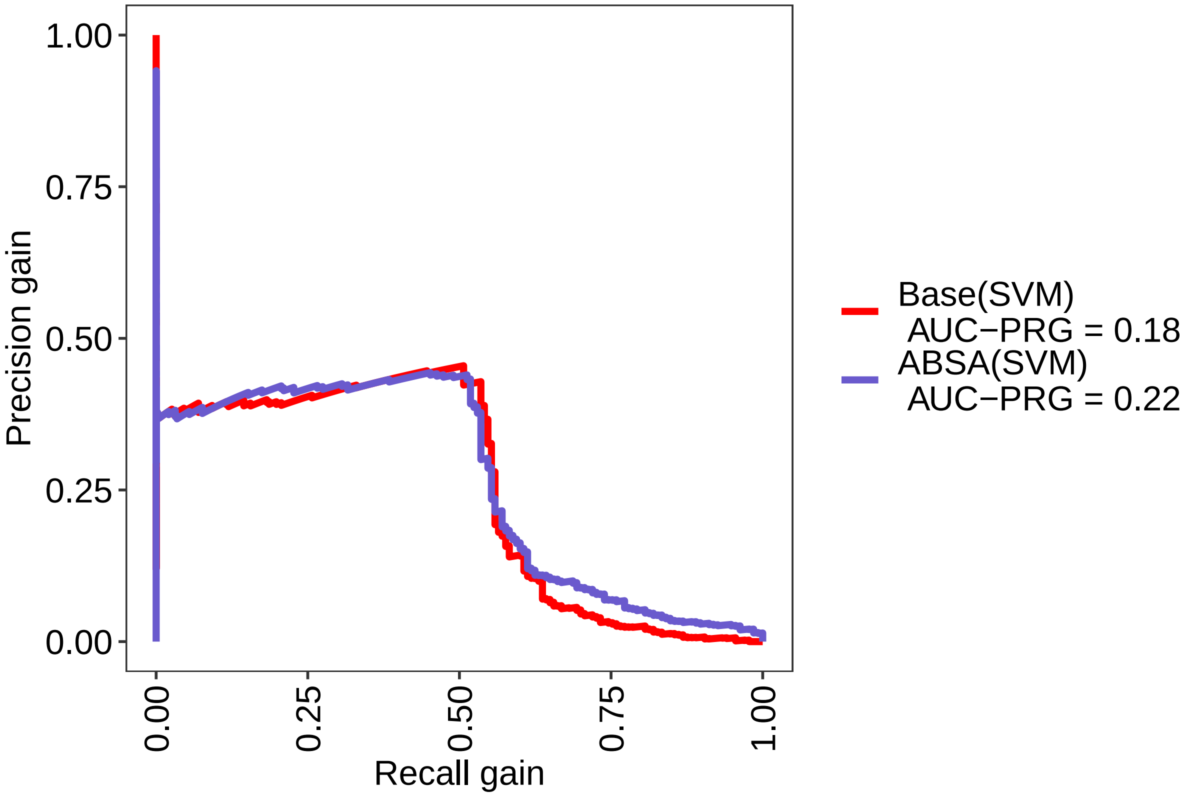
```
library(MLeval)
res <- evalm(list(Base_SVM,ABSA_SVM),gnames= c('Base(SVM)','ABSA(SVM)'))
```

```
## ***MLeval: Machine Learning Model Evaluation***

## Input: caret train function object

## Averaging probs.

## Group 1 type: repeatedcv

## Group 2 type: repeatedcv

## Observations: 1600

## Number of groups: 2

## Observations per group: 800

## Positive: Not_like
```

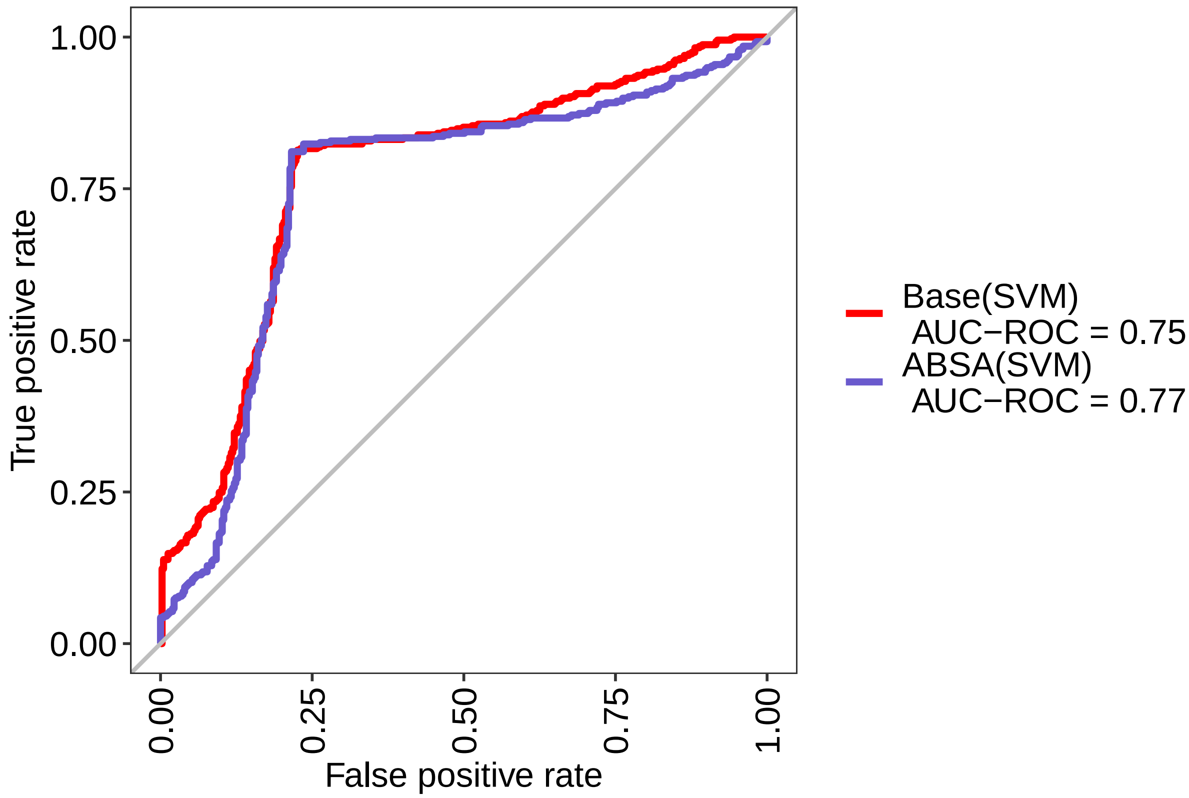
```
## Negative: Like

## Group: Base(SVM)

## Positive: 397

## Negative: 403

## Group: ABSA(SVM)

## Positive: 397

## Negative: 403

## ***Performance Metrics***
```




1.00
0.75
0.50
0.25
0.00
Precision
0.00
0.25
0.50
0.75
1.00
Recall(sensitivity)
Base(SVM)
AUC−PR = 0.7
ABSA(SVM)
AUC−PR = 0.75




```
## Base(SVM) Optimal Informedness = 0.595202230125444

## ABSA(SVM) Optimal Informedness = 0.587795563500447

## Base(SVM) AUC-ROC = 0.75

## ABSA(SVM) AUC-ROC = 0.77
```



4.3 Save result

```
result_top_sen <- data_top_sen_test%>%
  mutate(pred = predict(ABSA_SVM, data_top_sen_test),review_id = df_top_sen[-idx,]$review_id)%>%
  left_join(review,by = 'review_id')
```

5 Archive (Do not use)

```
# X_sen <- as.data.frame(as.matrix(df_sen%>%ungroup()%>%select(sentiment,0)))
# predictors <- names(X_sen)[!(names(X_sen) %in% "Liked")]
#
# Y_sen <- as.data.frame(as.matrix(df_sen%>%ungroup()%>%select(Liked)))
# Y_sen$Liked <- as.factor(ifelse(Y_sen$Liked == "1","Like","Not_like"))
# SVMmodel(X_sen,Y_sen)
#
# X_top_sen <- as.data.frame(as.matrix(df_top_sen%>%ungroup()%>%select(-review_id,-Liked)))
# Y_top_sen <- as.data.frame(as.matrix(df_top_sen%>%ungroup()%>%select(Liked)))
# Y_top_sen$Liked <- as.factor(ifelse(Y_top_sen$Liked == "1","Like","Not_like"))
# SVMmodel(X_top_sen,Y_top_sen)
```

```r
data <- as.data.frame(as.matrix(df_sen))%>%ungroup%>%select(-review_id)
data$Liked <- as.factor(ifelse(data$Liked == "1","Like","Not_like"))
# Split the dataset into training and testing sets
set.seed(8882)
idx <- sample(1:nrow(data), size = 0.7 * nrow(data), replace = FALSE)
data_train<-data[idx,]
data_test<-data[-idx,]

ctrl <- trainControl(method="repeatedcv",   # 10fold cross validation
                     repeats=5,         # do 5 repetitions of cv
                     summaryFunction=twoClassSummary,   # Use AUC to pick the best model
                     classProbs=TRUE,
                     savePredictions = T)

# run SVM
svm_Base <- train(Liked~.,
                   data = data,
                   method = "svmRadial",tuneLength = 5, # 5 values of the cost function
                   preProc = c("center","scale"),  # Center and scale data
                   metric="ROC",
                  trControl=ctrl)

# make predictions on the test set
pred <- predict(svm_Base, data_test)
# evaluate performance
confusionMatrix(pred, data_test$Liked)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Like Not_like
##   Like      120       33
##   Not_like   41      106
##
##                Accuracy : 0.7533
##                  95% CI : (0.7005, 0.8011)
##     No Information Rate : 0.5367
##     P-Value [Acc > NIR] : 8.027e-15
##
##                   Kappa : 0.5059
##
##  Mcnemar's Test P-Value : 0.4158
##
##             Sensitivity : 0.7453
##             Specificity : 0.7626
##          Pos Pred Value : 0.7843
##          Neg Pred Value : 0.7211
##              Prevalence : 0.5367
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.5100
##       Balanced Accuracy : 0.7540
##
##        'Positive' Class : Like
```

```
##
```

```r
data <- as.data.frame(as.matrix(df_top_sen))%>%ungroup%>%select(-review_id)
data$Liked <- as.factor(ifelse(data$Liked == "1","Like","Not_like"))
# Split the dataset into training and testing sets
set.seed(8882)
idx <- sample(1:nrow(data), size = 0.7 * nrow(data), replace = FALSE)
data_train<-data[idx,]
data_test<-data[-idx,]

ctrl <- trainControl(method="repeatedcv",   # 10fold cross validation
                     repeats=5,         # do 5 repetitions of cv
                     summaryFunction=twoClassSummary,   # Use AUC to pick the best model
                     classProbs=TRUE,
                     savePredictions = T)

# run SVM
svm_ABSA <- train(Liked~.,
                   data = data,
                   method = "svmRadial",tuneLength = 5, # 5 values of the cost function
                   preProc = c("center","scale"),  # Center and scale data
                   metric="ROC",
                  trControl=ctrl)

# make predictions on the test set
pred <- predict(svm_ABSA, data_test)
# evaluate performance
confusionMatrix(pred, data_test$Liked)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Like Not_like
##   Like      121       31
##   Not_like   40      108
##
##                Accuracy : 0.7633
##                  95% CI : (0.7111, 0.8103)
##     No Information Rate : 0.5367
##     P-Value [Acc > NIR] : 3.969e-16
##
##                   Kappa : 0.5262
##
##  Mcnemar's Test P-Value : 0.3424
##
##             Sensitivity : 0.7516
##             Specificity : 0.7770
##          Pos Pred Value : 0.7961
##          Neg Pred Value : 0.7297
##              Prevalence : 0.5367
##          Detection Rate : 0.4033
##    Detection Prevalence : 0.5067
##       Balanced Accuracy : 0.7643
##
```

```
##        'Positive' Class : Like
##
```