Homework 4

PSTAT 134/234

Table of contents

Homework 4

Note: If this is one of your two late homework submissions, please indicate below; also indicate whether it is your first or second late submission.

This homework assignment has you practice working with some text data, doing some natural language processing. I strongly advise using Lab 7 for assistance.

You also may need to use other functions. I encourage you to make use of our textbook(s) and use the Internet to help you solve these problems. You can also work together with your classmates. If you do work together, you should provide the names of those classmates below.

Names of Collaborators (if any): William Mahnke

Natural Language Processing

We'll work with the data in `data/spotify-review-data.csv`. This CSV file contains a total of 51,473 rows, each representing a unique user review for the Spotify application. The dataset has two columns:

- Review: This column contains the text of user reviews, reflecting their experiences, opinions, and feedback on the Spotify app.
- Sentiment label: This column categorizes each review as either "POSITIVE" or "NEGATIVE" based on its sentiment.

The data comes from this source at Kaggle: https://www.kaggle.com/datasets/alexandrakim2201/spotify-dataset

```
# data manipulation
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becon
```

```
library(tidymodels)
```

```
-- Attaching packages -------------------------------------- tidymodels 1.2.0 --
v broom        1.0.6     v rsample      1.2.1
v dials        1.3.0     v tune         1.2.1
v infer        1.0.7     v workflows    1.1.4
v modeldata    1.4.0     v workflowsets 1.1.0
v parsnip      1.2.1     v yardstick    1.3.1
v recipes      1.1.0
-- Conflicts ----------------------------------------- tidymodels_conflicts() --
x scales::discard() masks purrr::discard()
x dplyr::filter()   masks stats::filter()
x recipes::fixed()  masks stringr::fixed()
```

```
x dplyr::lag()      masks stats::lag()
x yardstick::spec() masks readr::spec()
x recipes::step()   masks stats::step()
* Learn how to get started at https://www.tidymodels.org/start/
```

```r
library(reshape2)
```

```
Attaching package: 'reshape2'

The following object is masked from 'package:tidyr':

    smiths
```

```r
# text mining
library(tidytext)
library(stringi)

# data visualization
library(ggplot2)
library(wordcloud)
```

```
Loading required package: RColorBrewer
```

```r
library(kableExtra)
```

```
Attaching package: 'kableExtra'

The following object is masked from 'package:dplyr':

    group_rows
```

```r
library(igraph)
```

```
Attaching package: 'igraph'

The following objects are masked from 'package:dials':
```

```
    degree, neighbors

The following objects are masked from 'package:lubridate':

    %--%, union

The following objects are masked from 'package:dplyr':

    as_data_frame, groups, union

The following objects are masked from 'package:purrr':

    compose, simplify

The following object is masked from 'package:tidyr':

    crossing

The following object is masked from 'package:tibble':

    as_data_frame

The following objects are masked from 'package:stats':

    decompose, spectrum

The following object is masked from 'package:base':

    union
```

```
library(ggraph)

# data modeling
library(tidymodels)
library(kernlab)
```

```
Attaching package: 'kernlab'

The following object is masked from 'package:scales':
```

```
    alpha

The following object is masked from 'package:purrr':

    cross

The following object is masked from 'package:ggplot2':

    alpha
```

```r
library(yardstick)

spotify <- read_csv('~/pstat-134-234/Homework/Homework 4/data/spotify-review-data.csv')
```

```
Rows: 52702 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): Review, label

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

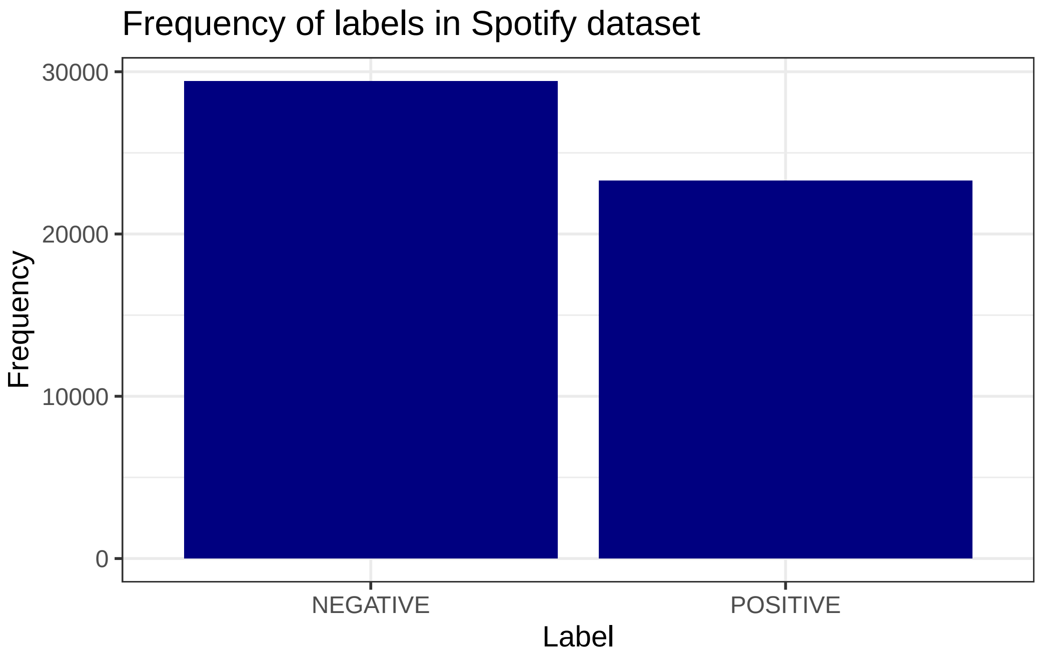
```r
theme_set(theme_bw())  # preset the bw theme
```

Exercise 1

Read the data into R (or Python, whichever you prefer).

Take a look at the distribution of `label`. Are there relatively even numbers of negative and positive reviews in the data set?

```r
ggplot(spotify) +
  geom_bar(aes(x= label), fill = "navy") +
  labs(title = "Frequency of labels in Spotify dataset",
       x = "Label",
       y = "Frequency")
```



The number of negative and positive reviews is relatively even.

Exercise 2

Take a random sample of $10,000$ reviews, stratified by `label`. All further exercises will be working with this smaller sample of reviews.

```
# stratify means to ensure that the distribution of the labels column in the sample is simila
set.seed(13)

spotify$id <- seq.int(nrow(spotify))
spotify_split <- initial_split(spotify, prop = 10001/nrow(spotify), strata = label)
spotify_train <- training(spotify_split)
spotify_test <- testing(spotify_split)

dim(spotify_train) # 10000 reviews
```

```
[1] 10000     3
```

Exercise 3

Tokenize the reviews into words.

Remove stop words. (You can use any pre-made list of stop words of your choice.)

Clean the reviews. Remove punctuation and convert the letters to lowercase.

Verify that this process worked correctly.

```
spotify_train$Review <- spotify_train$Review %>%
  iconv(from = 'UTF-8', to = 'ASCII//TRANSLIT') %>% # convert accented characters into ASCI c
  stri_replace_all_regex("[^\\p{L}\\p{N}\\s]", "") %>%  # remove everything except letter, nu
  tolower()
```

```
# data("stop_words") is SMART stopwords
spotify_train_words <- spotify_train %>%
  unnest_tokens(word, Review)  %>% # tokenize reviews into words
  anti_join(stop_words) # remove stop words
```

```
Joining with `by = join_by(word)`
```

```
spotify_train_words %>%
  count(word, sort = T) %>%
  head(n = 30) %>%
  kbl()
```

word	n
app	5180
music	3730
songs	2641
spotify	2639
song	2119
play	1677
love	1399
premium	1321
listen	1312
ads	1113
playlist	1050
playing	997
dont	990
im	929
update	835
fix	683
time	683
doesnt	584
phone	583
listening	573
ive	556
podcasts	478
NA	472
stop	456
free	446
playlists	432
annoying	409
wont	380
easy	373
pay	368

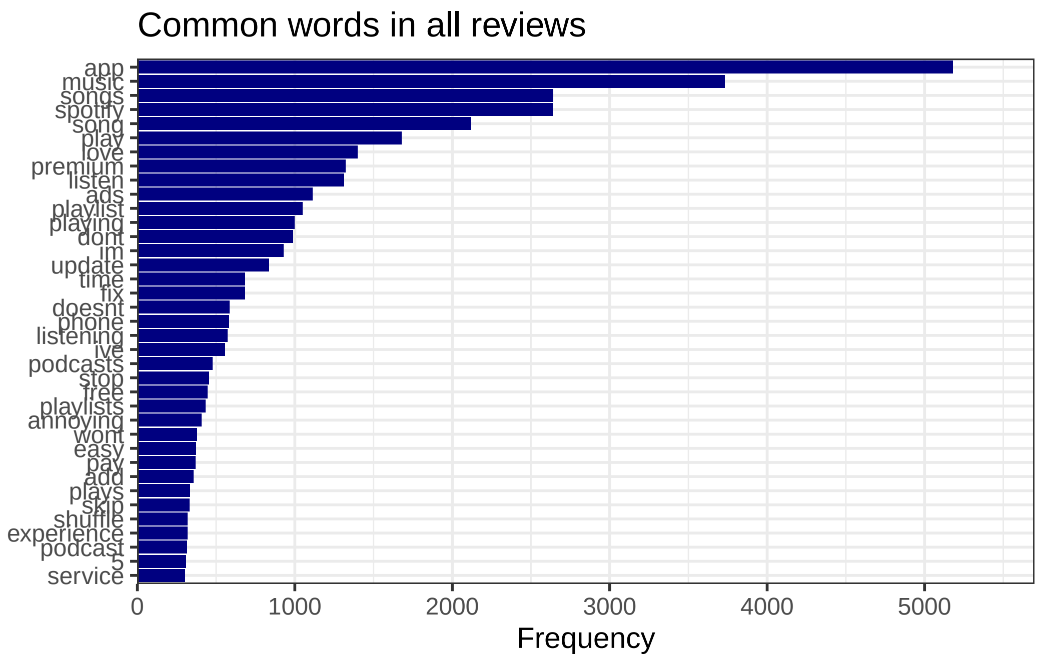
Exercise 4

Create a bar chart of the most commonly-occurring words (not including stop words).

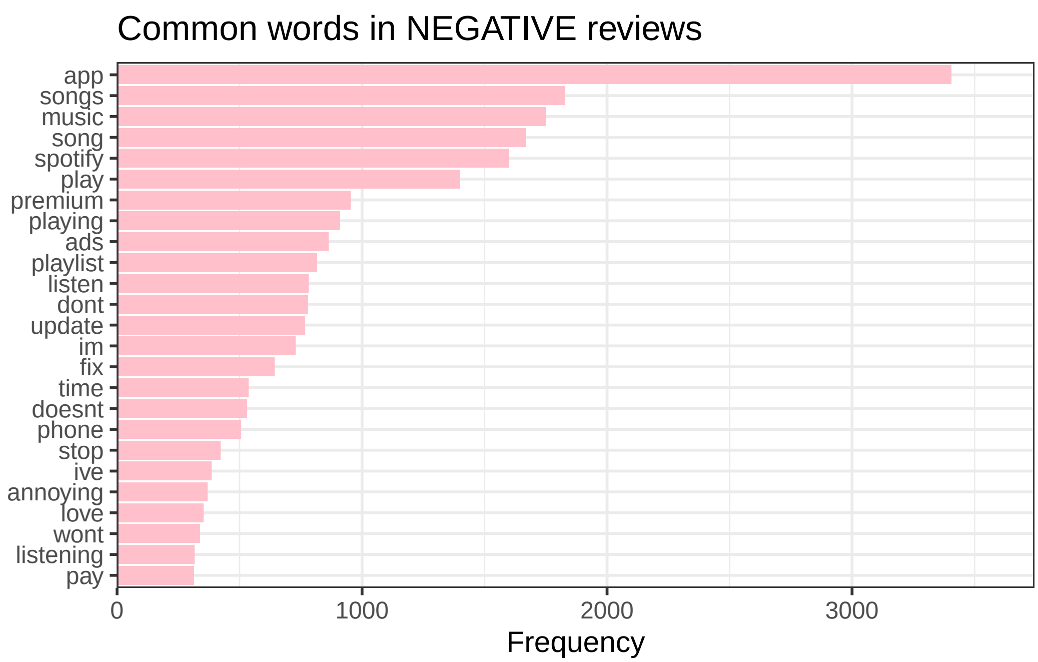
Create bar charts of the most commonly-occurring words, broken down by `label`. What words are more common in positive reviews? What words are more common in negative reviews?

```
spotify_train_words %>%
  filter(!is.na(word)) %>%
```

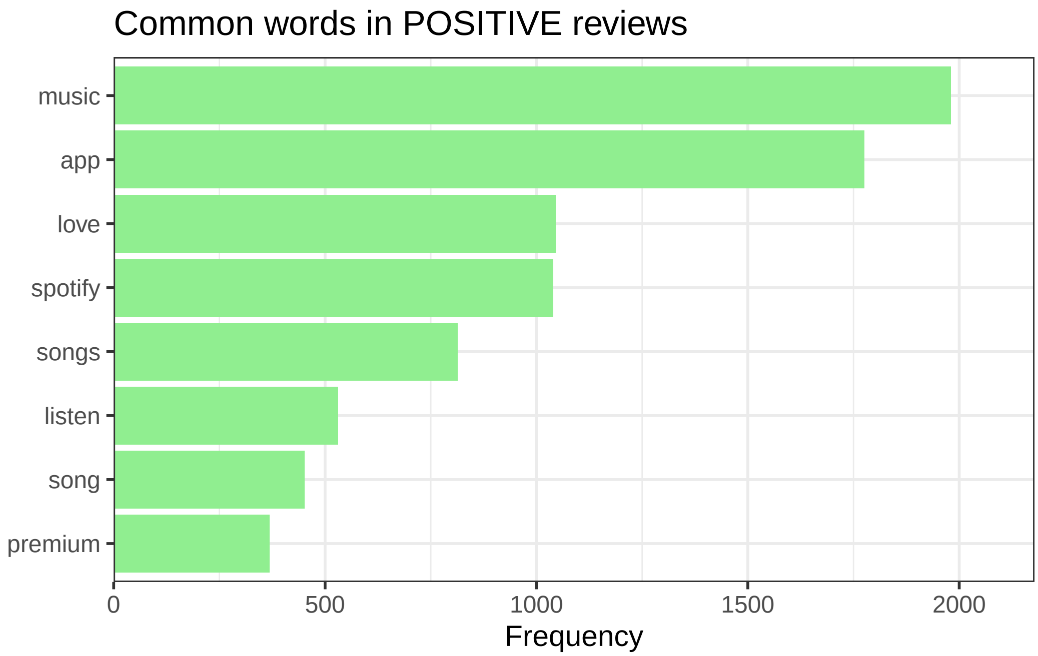
```
  count(word, sort = TRUE) %>%
  filter(n > 300) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col(fill = "navy") +
  scale_x_continuous(expand = expansion(mult = c(0, .1))) +
  labs(title = "Common words in all reviews", x = "Frequency", y = NULL)
```




```
spotify_train_words %>%
  filter(label == 'NEGATIVE') %>%
  count(word, sort = TRUE) %>%
  filter(n > 300) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col(fill = "pink") +
  scale_x_continuous(expand = expansion(mult = c(0, .1))) +
  labs(title = "Common words in NEGATIVE reviews", x = "Frequency", y = NULL)
```




```
spotify_train_words %>%
  filter(label == 'POSITIVE') %>%
  filter(!is.na(word)) %>%
  count(word, sort = TRUE) %>%
  filter(n > 300) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col(fill = "lightgreen") +
  scale_x_continuous(expand = expansion(mult = c(0, .1))) +
  labs(title = "Common words in POSITIVE reviews", x = "Frequency", y = NULL)
```



Some of the most frequent words in all reviews are: "app", "songs", "music", "spotify," and "play". These words are used in both positive and negative reviews. The words more specific to negative reviews are: "playlist", "ads", "update", "fix", time", and "annoying". A word specific to positive reviews is "love". "There are more words used in negative reviews that have a frequency of over 300.

Exercise 5

Create a word cloud of the most commonly-occurring words overall, broken down by "positive" or "negative" sentiment (using the Bing sentiment lexicon).

```
spotify_train_words %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("red", "blue"),
                   max.words = 100)
```

```
Joining with `by = join_by(word)`
```

Exercise 6

Calculate the tf-idf values for the words in the dataset.

Find the 30 words with the largest tf-idf values.

Find the 30 words with the smallest tf-idf values.

```
spotify_train_tf_idf <- spotify_train_words %>%
  count(id, word) %>%
  bind_tf_idf(term = word,
              document = id,
              n = n)

# 30 largest tf-idf values
spotify_train_tf_idf %>%
  arrange(desc(tf_idf)) %>%
  head(n = 30) %>%
  kbl() %>%
  scroll_box(width = "400px", height = "500px")
```

id	word	n	tf	idf	tf_idf
47425	advertisemens	1	1	9.200290	9.200290
49137	topical	1	1	9.200290	9.200290
49152	unprecedented	1	1	9.200290	9.200290
49309	ilkee	1	1	9.200290	9.200290
49377	neverminds	1	1	9.200290	9.200290
49450	likeeeit	1	1	9.200290	9.200290
49622	2023	1	1	9.200290	9.200290
49720	mzemer	1	1	9.200290	9.200290
50179	goooddd	1	1	9.200290	9.200290
50262	lovee	1	1	9.200290	9.200290
50264	sexy	1	1	9.200290	9.200290
50314	manigandn	1	1	9.200290	9.200290
50676	superpower	1	1	9.200290	9.200290
50792	temaa	1	1	9.200290	9.200290
50830	loveitt	1	1	9.200290	9.200290
51008	exelent	1	1	9.200290	9.200290
51197	goool	1	1	9.200290	9.200290
51525	muslim	1	1	9.200290	9.200290
51860	peermohamd	2	1	9.200290	9.200290
51902	momment	1	1	9.200290	9.200290
52025	ape	1	1	9.200290	9.200290
52409	ankushpal	1	1	9.200290	9.200290
52503	excelente	1	1	9.200290	9.200290
49448	goood	1	1	8.507143	8.507143
49677	sensational	1	1	8.507143	8.507143
49812	amaze	1	1	8.507143	8.507143
50191	likeee	1	1	8.507143	8.507143
50237	likeee	1	1	8.507143	8.507143
50798	marvelous	1	1	8.507143	8.507143
50988	satisfy	1	1	8.507143	8.507143

```r
# 30 smallest tf-idf values
spotify_train_tf_idf %>%
  arrange(tf_idf) %>%
  head(n = 30) %>%
  kbl() %>%
  scroll_box(width = "400px", height = "500px")
```

id	word	n	tf	idf	tf_idf
12625	app	1	0.0181818	0.9313018	0.0169328
14808	app	1	0.0222222	0.9313018	0.0206956
38943	app	1	0.0232558	0.9313018	0.0216582
10814	app	1	0.0238095	0.9313018	0.0221739
4566	app	1	0.0243902	0.9313018	0.0227147
9951	app	1	0.0243902	0.9313018	0.0227147
27395	app	1	0.0243902	0.9313018	0.0227147
31460	app	1	0.0243902	0.9313018	0.0227147
31936	app	1	0.0243902	0.9313018	0.0227147
39921	app	1	0.0243902	0.9313018	0.0227147
45589	app	1	0.0243902	0.9313018	0.0227147
12070	app	1	0.0250000	0.9313018	0.0232825
28506	app	1	0.0250000	0.9313018	0.0232825
34036	app	1	0.0250000	0.9313018	0.0232825
37032	app	1	0.0250000	0.9313018	0.0232825
38787	app	1	0.0250000	0.9313018	0.0232825
47525	app	1	0.0250000	0.9313018	0.0232825
19880	app	1	0.0256410	0.9313018	0.0238795
20446	app	1	0.0256410	0.9313018	0.0238795
40031	app	1	0.0256410	0.9313018	0.0238795
43523	app	1	0.0256410	0.9313018	0.0238795
46987	app	1	0.0256410	0.9313018	0.0238795
894	app	1	0.0263158	0.9313018	0.0245079
10801	app	1	0.0263158	0.9313018	0.0245079
19366	app	1	0.0263158	0.9313018	0.0245079
26026	app	1	0.0263158	0.9313018	0.0245079
27473	app	1	0.0263158	0.9313018	0.0245079
39309	app	1	0.0263158	0.9313018	0.0245079
39590	app	1	0.0263158	0.9313018	0.0245079
5294	app	1	0.0270270	0.9313018	0.0251703

Exercise 7

Find the 30 most commonly occuring bigrams.

Create graphs visualizing the networks of bigrams, broken down by `label`. That is, make one graph of the network of bigrams for the positive reviews, and one graph of the network for the negative reviews.

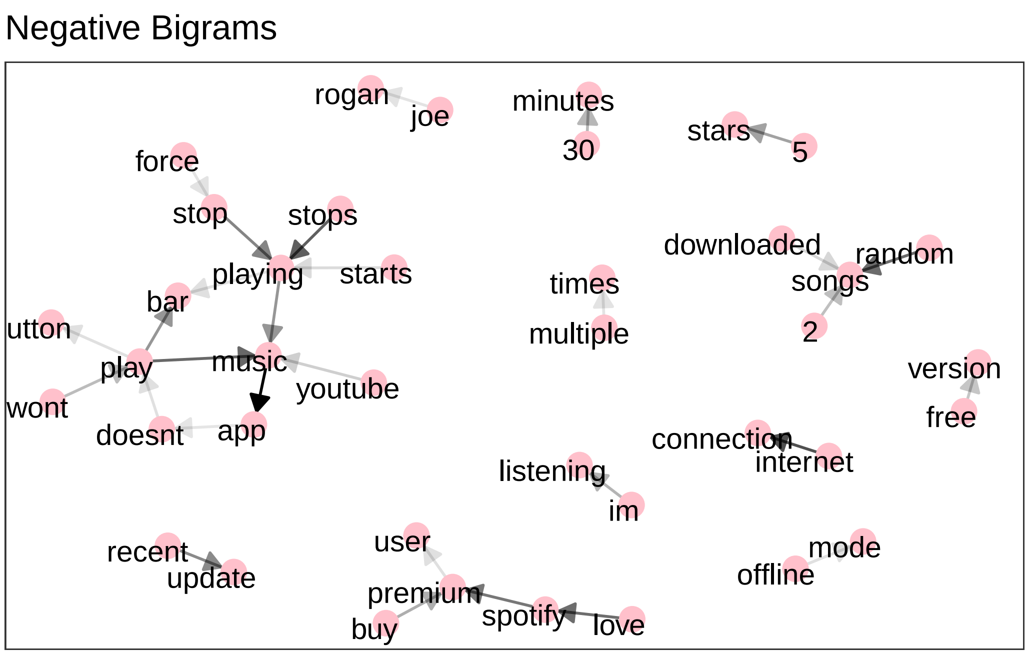
What patterns do you notice?

```
spotify_train_bigrams <- spotify_train %>%
  unnest_tokens(bigram, Review, token = 'ngrams', n =2) %>% # create bigrams (pairs of words
  separate(bigram, c("word1", "word2"), sep = " ") %>% # split bigram into sep words
  filter(!word1 %in% stop_words$word) %>% # filter for stop words
  filter(!word2 %in% stop_words$word) %>%
  unite(bigram, word1, word2, sep = " ") # combine words into bigram

spotify_train_bigrams %>%
  count(bigram, sort = TRUE) %>%
  slice(-1) %>%
  head(n = 30) %>%
  kbl() %>%
  scroll_box(width = "400px", height = "500px")
```

bigram	n
music app	351
love spotify	217
spotify premium	103
5 stars	101
play music	100
internet connection	96
random songs	89
music streaming	87
stops playing	87
stop playing	75
recent update	74
playing music	73
nice app	72
play bar	67
free version	66
buy premium	64
30 minutes	62
sound quality	62
im listening	60
joe rogan	60
wont play	57
amazing app	56
youtube music	56
5 star	55
favorite music	54
2 songs	53
favorite songs	49
play songs	49
premium user	48
starts playing	48

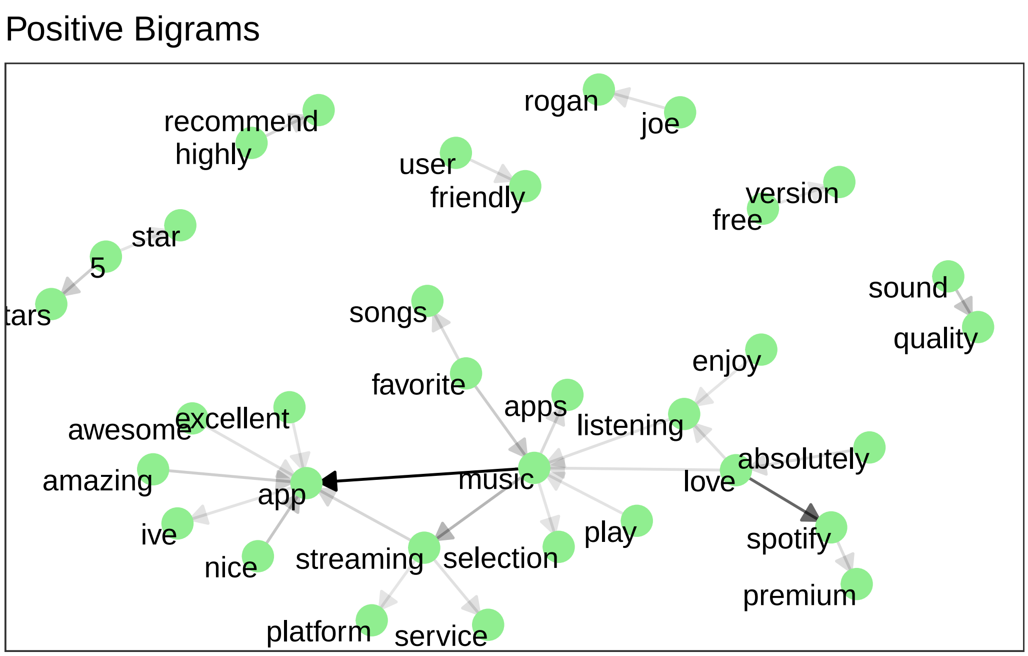
```
bigram_graph_neg <- spotify_train_bigrams %>%
  filter(label == "NEGATIVE") %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  count(word1, word2) %>%
  arrange(desc(n)) %>%
  slice(-1) %>%
  slice_head(n = 30) %>%
 # mutate(n = as.integer(n)) %>%
  graph_from_data_frame()
```

```r
bigram_graph_pos <- spotify_train_bigrams %>%
  filter(label == "POSITIVE") %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  count(word1, word2) %>%
  arrange(desc(n)) %>%
  slice(-1) %>%
  slice_head(n = 30) %>%
 # mutate(n = as.integer(n)) %>%
  graph_from_data_frame()

a <- grid::arrow(type = "closed", length = unit(.095, "inches"))

ggraph(bigram_graph_neg, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.07, 'inches')) +
  geom_node_point(color = "pink", size = 4) +
  geom_node_text(aes(label = name), vjust = 0.75, hjust = 0.75) +
  ggtitle(label = "Negative Bigrams")
```




```r
ggraph(bigram_graph_pos, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.07, 'inches')) +
```

```
  geom_node_point(color = "lightgreen", size = 5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
  ggtitle(label = "Positive Bigrams")
```



"Music app", "love spotify" and "spotify premium" are common bigrams (which appear in over 100 reviews). The common bigrams are all common phrases that people would write when describing Spotify. In negative reviews, people would often precede the word "play" with "won't", "doesn't", or "button", suggesting there are issues with how these users are unable to play their music. Other phrases in negative reviews are "30 minutes" and "free version" suggesting people are upset about the non-premium version of the app. In the positive reviews, people typically describe "music" with "favorite" and "love", because they are happy by the performance of the app. Other phrases are: "highly recommend", "user friendly", and "quality sound."

Exercise 8

Using the tokenized **words** and their corresponding tf-idf scores, fit a **linear support vector machine** to predict whether a given review is positive or negative.

- Split the data using stratified sampling, with 70% training and 30% testing;
- Drop any columns with zero variance;
- Fit a linear support vector machine using default values for any hyperparameters;

- Calculate the model **accuracy** on your testing data.

```
spotify_tokens <- spotify_train_tf_idf %>%
  rename(review_label = label,
         review_id = id) %>%
  group_by(review_id, review_label, word) %>%
  summarise(tf_idf = sum(tf_idf), .groups = 'drop') %>%
  pivot_wider(names_from = word,
              values_from = tf_idf,
              values_fill = 0) %>%
  mutate(review_label = factor(review_label)) %>%
  select(-review_id)

spotify_tokens_split <- initial_split(spotify_tokens, prop = 0.7)
spotify_tokens_train <- training(spotify_tokens_split)
spotify_tokens_test <- testing(spotify_tokens_split)

# recipe
spotify_recipe <- recipe(review_label ~ ., data = spotify_tokens_train) %>%
  step_zv(all_predictors())

prep(spotify_recipe) %>%
  bake(spotify_tokens_train) %>%
  select(review_label, everything()) %>%
  summary()

svmlin <- svm_linear(mode = "classification", cost = 1, margin = 0.1) %>%
  set_engine("kernlab")

svmlin_wkflow <- workflow() %>%
  add_model(svmlin) %>%
  add_recipe(spotify_recipe)

svmlin_fit <- svmlin_wkflow %>%
  fit(data = spotify_tokens_train)

svmlin_pred <- predict(svmlin_fit, new_data = spotify_tokens_test)

# to avoid fitting the model again on the training data, i saved the model as R object to loa
save(svmlin_fit, file = "~/pstat-134-234/Homework/Homework 4/svmlin_fit.RData")
save(spotify_tokens_train, file = "~/pstat-134-234/Homework/Homework 4/spotify_tokens_train.R
save(spotify_tokens_test, file = "~/pstat-134-234/Homework/Homework 4/spotify_tokens_test.RDa
```

```r
save(svmlin_fit, file = "~/pstat-134-234/Homework/Homework 4/svmlin_fit.RData")
save(svmlin_pred, file = "~/pstat-134-234/Homework/Homework 4/svmlin_pred.RData")
```

```r
load('~/pstat-134-234/Homework/Homework 4/spotify_tokens_test.RData')
load('~/pstat-134-234/Homework/Homework 4/spotify_tokens_train.RData')

# svmlin_pred <- predict(svmlin_fit, new_data = spotify_tokens_test)
load('~/pstat-134-234/Homework/Homework 4/svmlin_pred.RData')

svmlin_results <- bind_cols(spotify_tokens_test %>% select(review_label),
                            svmlin_pred)

# model accuracy of 73.44%
accuracy(svmlin_results, truth = review_label, estimate = .pred_class)
```

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.734
```

The model is 73.44% accurate when predicting reviews on the testing set.

For 234 Students

Exercise 9

Using **either** Bag of Words or Word2Vec, extract a matrix of features. (Note: You can reduce the size of the dataset even further by working with a sample of $3,000$ reviews if need be.)

Exercise 10

Fit and tune a **logistic regression model, using lasso regularization**. Follow the same procedure as before, with a few changes:

- Stratified sampling, with a 70/30 split;
- Drop any columns with zero variance;
- Tune `penalty`, using the default values;
- Calculate your best model's **accuracy** on the testing data.