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):

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

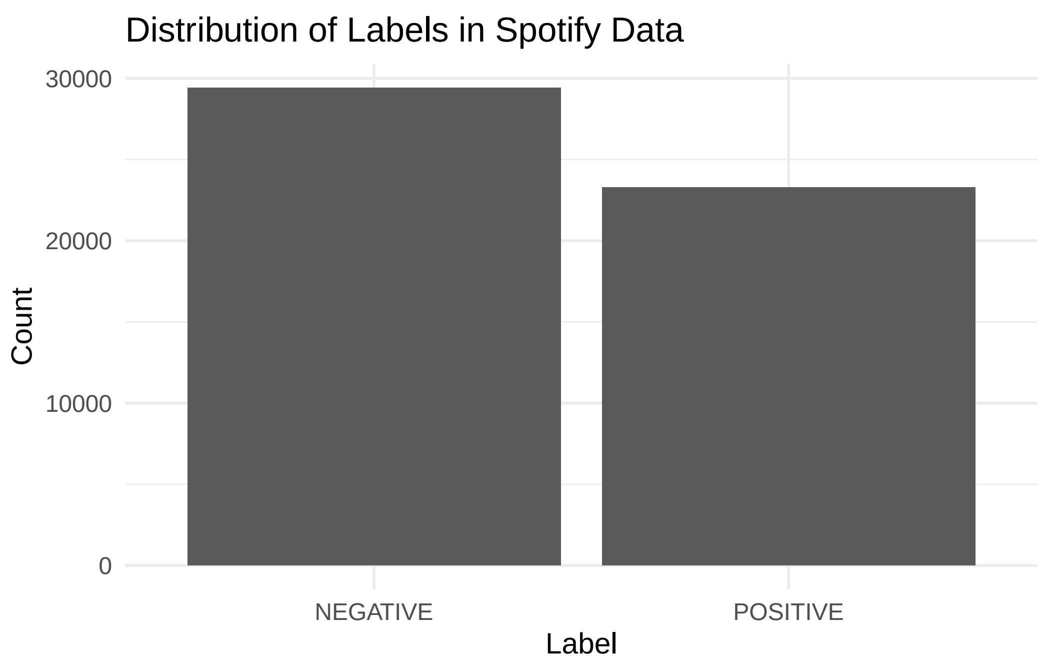
Exercise 1

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
library(tidyverse)
library(tidymodels)
library(reshape2)
library(wordcloud)
library(ggraph)
library(tidytext)
library(httr)
library(igraph)
library(data.table)
library(textdata)
library(ggplot2)
library(ggrepel)
library(plotly)
library(umap)
library(word2vec)
library(tm)
library(kableExtra)
library(LiblineaR)

spotify <- read.csv("data/spotify-review-data.csv")
ggplot(spotify, aes(x = label)) +
```

```
  geom_bar() +
  labs(title = "Distribution of Labels in Spotify Data",
       x = "Label",
       y = "Count") +
  theme_minimal()
```




```
spotify$id <- seq.int(nrow(spotify))
spotify_for_later <- spotify
```

The number of negative reviews 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.

```
spotify_sample <- spotify %>%
  group_by(label) %>%
  sample_frac(size = 10000 / nrow(spotify), replace = F) %>%
  ungroup()

# prop.table(table(spotify_sample$label))
```

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.

```r
spotify_sample %>%
  unnest_tokens(word, Review) %>%
  head(10)
```

```
# A tibble: 10 x 3
   label       id word
   <chr>    <int> <chr>
 1 NEGATIVE 22279 great
 2 NEGATIVE 22279 selection
 3 NEGATIVE 22279 and
 4 NEGATIVE 22279 integrations
 5 NEGATIVE 22279 point
 6 NEGATIVE 22279 loss
 7 NEGATIVE 22279 for
 8 NEGATIVE 22279 ui
 9 NEGATIVE 22279 though
10 NEGATIVE 22279 could
```

```r
stop_words %>%
  head(n = 10)
```

```
# A tibble: 10 x 2
   word        lexicon
   <chr>       <chr>
 1 a           SMART
 2 a's         SMART
 3 able        SMART
 4 about       SMART
 5 above       SMART
 6 according   SMART
 7 accordingly SMART
 8 across      SMART
 9 actually    SMART
```

```
10 after       SMART
```

```r
# token and no stop words
spotify_sample %>%
  filter(!is.na(Review)) %>%
  unnest_tokens(word, Review) %>%
  anti_join(stop_words) %>%
  count(word, sort = T)
```

```
# A tibble: 8,995 x 2
   word        n
   <chr>   <int>
 1 app      5564
 2 music    4042
 3 spotify  2862
 4 songs    2737
 5 song     2185
 6 play     1695
 7 listen   1420
 8 love     1404
 9 premium  1306
10 ads      1274
# i 8,985 more rows
```

```r
head(spotify_sample, 10)
```

```
# A tibble: 10 x 3
   Review                                                       label    id
   <chr>                                                        <chr> <int>
 1 Great selection and integrations. Point loss for UI though. Coul~ NEGA~ 22279
 2 New update is the worst. I'll be listening to a playlist when I ~ NEGA~ 31212
 3 Update- love spotify in general but the app is super glitchy on ~ NEGA~ 14885
 4 Can there be any other app that consistently makes the user expe~ NEGA~ 30987
 5 Too many adds. Constant issues with skipping to next song. Pract~ NEGA~  9614
 6 The app is full of bugs now, it keeps skipping and crashing. The~ NEGA~ 36695
 7 The app only works half the time, I've got it connected to my al~ NEGA~ 24973
 8 App is trash.. If you can use podcast addict do it. They are awe~ NEGA~  4678
 9 It used to be fun and all but now it doesnt let me play my playl~ NEGA~ 38670
10 The app was working fine until a couple of weeks ago. I can play~ NEGA~  7968
```

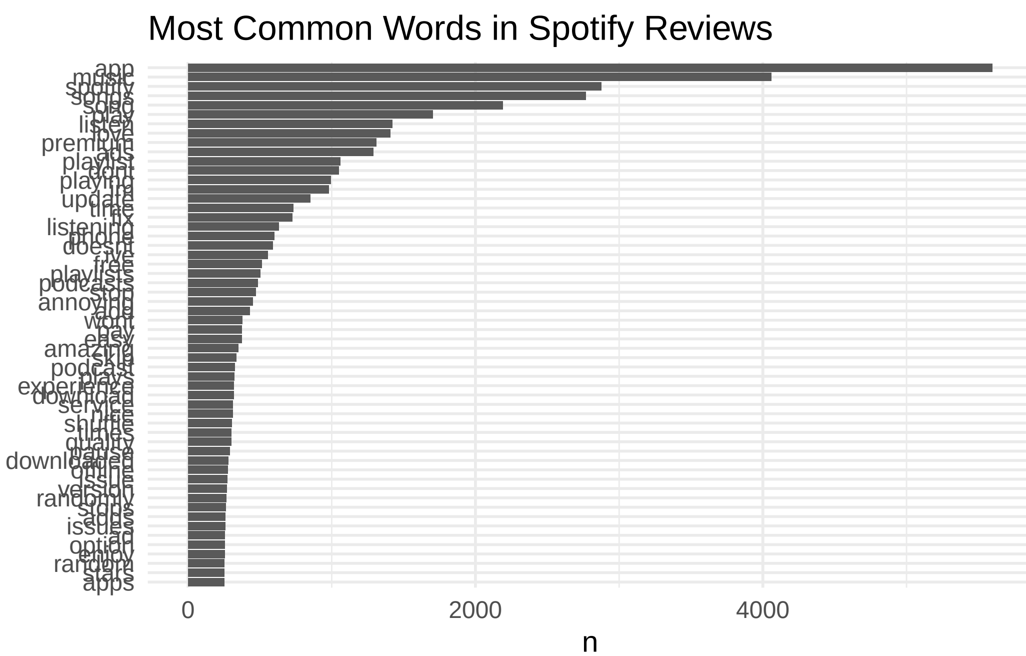
```r
# removing HTML tags, replacing with a space
spotify_sample$Review <- str_replace_all(spotify_sample$Review, pattern = "<.*?>", " ")
# removing "\n", replacing with a space
spotify_sample$Review <- str_replace_all(spotify_sample$Review, pattern = "\n", " ")
# removing "&amp;" and "&gt;"
spotify_sample$Review <- str_replace_all(spotify_sample$Review, pattern = "&amp;", " ")
spotify_sample$Review <- str_replace_all(spotify_sample$Review , pattern = "&gt;", " ")

remove <- c('\n',
            '[[:punct:]]',
            'nbsp',
            '[[:digit:]]',
            '[[:symbol:]]',
            '^br$',
            'href',
            'ilink') %>%
  paste(collapse = '|')
# removing any other weird characters,
# any backslashes, adding space before capital
# letters and removing extra whitespace,
# replacing capital letters with lowercase letters
spotify_sample$Review <- spotify_sample$Review %>%
  str_remove_all('\'') %>%
  str_replace_all(remove, ' ') %>%
  str_replace_all("([a-z])([A-Z])", "\\1 \\2") %>%
  tolower() %>%
  str_replace_all("â|ï|ð|ÿ|œ|ž|š|^", " ") %>%
  str_replace_all("\\s+", " ") %>%
  str_trim()

# take a look at random row
spotify_sample$Review[66:69]
```

```
[1] "i dont really like the new update i think its unfair to only have skips per hour cause i
[2] "this app sucks they add songs to my albums that i dont want like i made an album from th
[3] "the april update is full of bugs playlists or albums suddenly stop without warning the a
[4] "used really enjoy the app but now the lay out is freaking ridiculous you dont need to be
```

Exercise 4

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

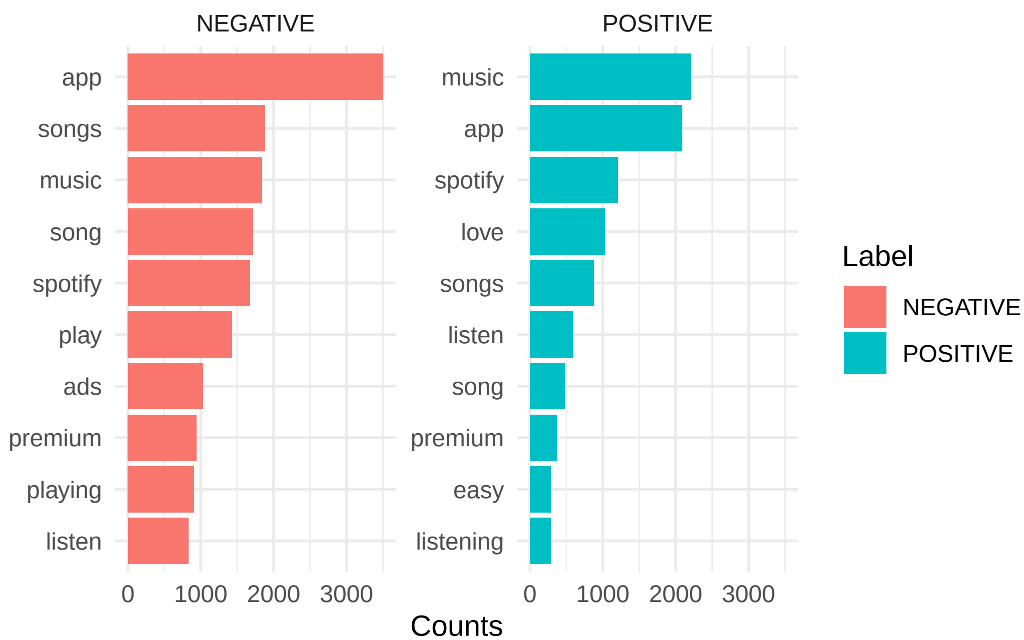
```
spotify_sample %>%
  unnest_tokens(word, Review) %>%
  anti_join(stop_words) %>%
  count(word, sort = T) %>%
  filter(n > 250) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL, title = "Most Common Words in Spotify Reviews") +
  theme_minimal()
```



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?

```
word_counts <- spotify_sample %>%
  unnest_tokens(word, Review) %>%
  anti_join(stop_words) %>%
  count(label, word, sort = T) %>%
  group_by(label) %>%
  slice_max(n, n = 10) %>%
  ungroup()
```

```
word_counts %>%
  mutate(word = reorder_within(word, n, label)) %>%
  ggplot(aes(n, word, fill = label)) +
  geom_col() +
  facet_wrap(~label, scales = "free_y") +
  scale_y_reordered() +
  labs(x = "Counts", y = NULL, fill = "Label") +
  theme_minimal()
```



“app” is most common in negative reviews, “music” is the most common words in positive reviews.

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_sample %>%
  unnest_tokens(word, Review) %>%
  anti_join(stop_words) %>%
  ungroup() %>%
  inner_join(get_sentiments("bing")) %>%
```

```
  count(word, sentiment, sort = T) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("red", "blue"), scale = c(4, 0.5), max.words = 70)
```

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.

```
tf_idf <- spotify_sample %>%
  unnest_tokens(word, Review) %>%
  anti_join(stop_words) %>%
  count(id, label, word) %>%
  bind_tf_idf(term = word,
              document = id,
              n = n)

tf_idf %>%
  arrange(desc(tf_idf)) %>%
  head(n = 30) %>%
```

```
kbl() %>%
add_header_above(c("Top 30 Words with the Largest TF-IDF Values" = 7)) %>%
scroll_box(width = "400px", height = "500px")
```

Top 30 Words with the Largest TF-IDF Values						
id	label	word	n	tf	idf	tf_idf
2409	POSITIVE	authentic	1	1	9.199987	9.199987
49152	POSITIVE	unprecedented	1	1	9.199987	9.199987
49309	POSITIVE	ilkee	1	1	9.199987	9.199987
49468	POSITIVE	manu	1	1	9.199987	9.199987
49887	POSITIVE	osam	1	1	9.199987	9.199987
50030	POSITIVE	sheesh	1	1	9.199987	9.199987
50526	POSITIVE	lovetr	1	1	9.199987	9.199987
50636	POSITIVE	gooooooooood	1	1	9.199987	9.199987
50792	POSITIVE	temaa	1	1	9.199987	9.199987
50819	POSITIVE	loveeees	1	1	9.199987	9.199987
50827	POSITIVE	ecstatic	1	1	9.199987	9.199987
50901	POSITIVE	loveitt	1	1	9.199987	9.199987
50995	POSITIVE	besttt	1	1	9.199987	9.199987
51008	POSITIVE	exelent	1	1	9.199987	9.199987
51191	POSITIVE	yayayayyayaayay	1	1	9.199987	9.199987
51260	POSITIVE	kkeep	1	1	9.199987	9.199987
51279	POSITIVE	exprience	1	1	9.199987	9.199987
51468	POSITIVE	complains	1	1	9.199987	9.199987
51696	POSITIVE	wowamazing	1	1	9.199987	9.199987
51958	POSITIVE	wooooow	1	1	9.199987	9.199987
52163	POSITIVE	shake	1	1	9.199987	9.199987
52532	POSITIVE	wark	1	1	9.199987	9.199987
52595	POSITIVE	briliant	1	1	9.199987	9.199987
52663	POSITIVE	weid	1	1	9.199987	9.199987
49812	POSITIVE	amaze	1	1	8.506840	8.506840
50175	POSITIVE	perfection	1	1	8.506840	8.506840
50873	POSITIVE	aps	1	1	8.506840	8.506840
51318	POSITIVE	amaze	1	1	8.506840	8.506840
52523	POSITIVE	sensational	1	1	8.506840	8.506840
52661	POSITIVE	congrats	1	1	8.506840	8.506840

```
tf_idf %>%
  arrange(tf_idf) %>%
  head(n = 30) %>%
```

```
kbl() %>%
add_header_above(c("Top 30 Words with the Smallest TF-IDF Values" = 7)) %>%
scroll_box(width = "400px", height = "500px")
```

Top 30 Words with the Smallest TF-IDF Values						
id	label	word	n	tf	idf	tf_idf
26365	NEGATIVE	app	1	0.0125000	0.8394476	0.0104931
26365	NEGATIVE	music	1	0.0125000	1.1089719	0.0138621
47123	NEGATIVE	app	1	0.0169492	0.8394476	0.0142279
24248	NEGATIVE	app	1	0.0181818	0.8394476	0.0152627
22509	POSITIVE	app	1	0.0212766	0.8394476	0.0178606
13813	POSITIVE	app	1	0.0222222	0.8394476	0.0186544
32991	NEGATIVE	app	1	0.0227273	0.8394476	0.0190784
35467	NEGATIVE	app	1	0.0232558	0.8394476	0.0195220
14808	NEGATIVE	app	1	0.0238095	0.8394476	0.0199868
28871	NEGATIVE	app	1	0.0238095	0.8394476	0.0199868
35795	NEGATIVE	app	1	0.0238095	0.8394476	0.0199868
19374	NEGATIVE	app	1	0.0243902	0.8394476	0.0204743
47525	NEGATIVE	app	1	0.0250000	0.8394476	0.0209862
2057	NEGATIVE	app	1	0.0256410	0.8394476	0.0215243
25246	NEGATIVE	app	1	0.0256410	0.8394476	0.0215243
43759	NEGATIVE	app	1	0.0256410	0.8394476	0.0215243
45589	POSITIVE	app	1	0.0256410	0.8394476	0.0215243
27576	NEGATIVE	app	1	0.0263158	0.8394476	0.0220907
36563	NEGATIVE	app	1	0.0263158	0.8394476	0.0220907
44186	NEGATIVE	app	1	0.0263158	0.8394476	0.0220907
4861	NEGATIVE	app	1	0.0270270	0.8394476	0.0226878
18226	NEGATIVE	app	1	0.0270270	0.8394476	0.0226878
28441	POSITIVE	app	1	0.0270270	0.8394476	0.0226878
29294	NEGATIVE	app	1	0.0270270	0.8394476	0.0226878
46019	NEGATIVE	app	1	0.0270270	0.8394476	0.0226878
3165	NEGATIVE	app	1	0.0277778	0.8394476	0.0233180
10293	NEGATIVE	app	1	0.0277778	0.8394476	0.0233180
16844	NEGATIVE	app	1	0.0277778	0.8394476	0.0233180
26384	NEGATIVE	app	1	0.0277778	0.8394476	0.0233180
27473	NEGATIVE	app	1	0.0277778	0.8394476	0.0233180

Exercise 7

Find the 30 most commonly occuring bigrams.

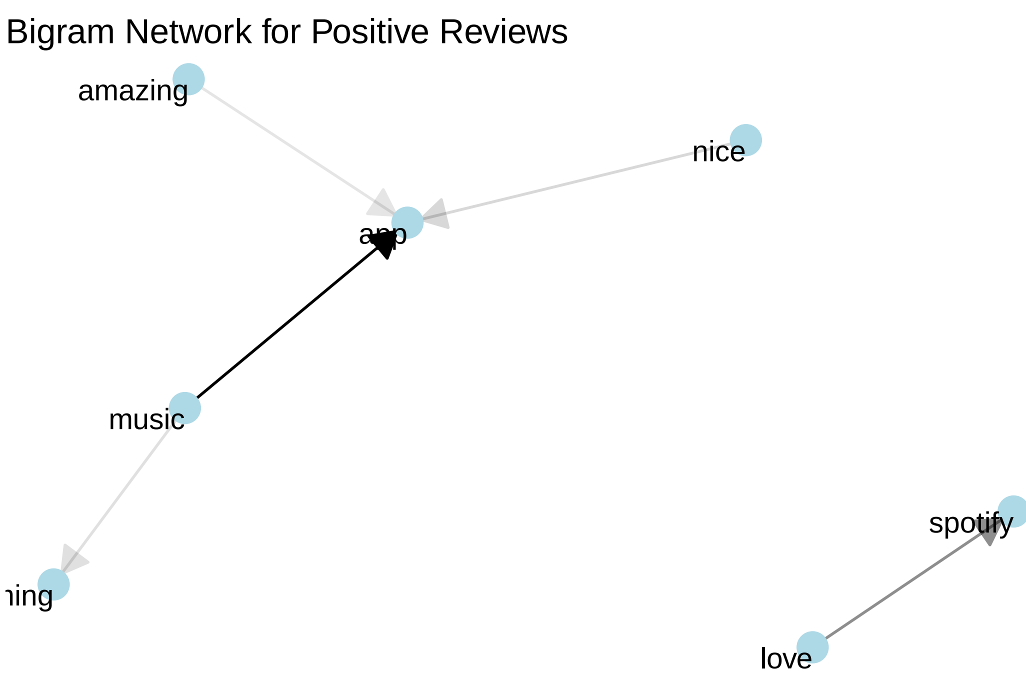
```r
nrc_bigrams <- spotify_sample %>%
  unnest_tokens(bigram, Review, token = "ngrams", n = 2) %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(!is.na(word1), !is.na(word2)) %>%
  unite(bigram, word1, word2, sep = " ")

nrc_bigrams %>%
  count(bigram, sort = T) %>%
  head(n = 30) %>%
  kbl() %>%
  add_header_above(c("Top 30 Most Common Bigrams" = 2)) %>%
  scroll_box(width = "400px", height = "500px")
```

Top 30 Most Common Bigrams	
bigram	n
music app	399
love spotify	233
spotify premium	125
internet connection	101
nice app	97
play music	97
music streaming	93
stops playing	86
free version	80
sound quality	79
random songs	78
amazing app	77
wont play	70
joe rogan	68
favorite songs	65
playing music	65
stop playing	64
recent update	63
downloaded songs	58
buy premium	56
music apps	56
premium user	56
streaming app	55
skip songs	53
tube music	53
play bar	51
spotify app	51
add songs	50
play pause	50
play songs	50

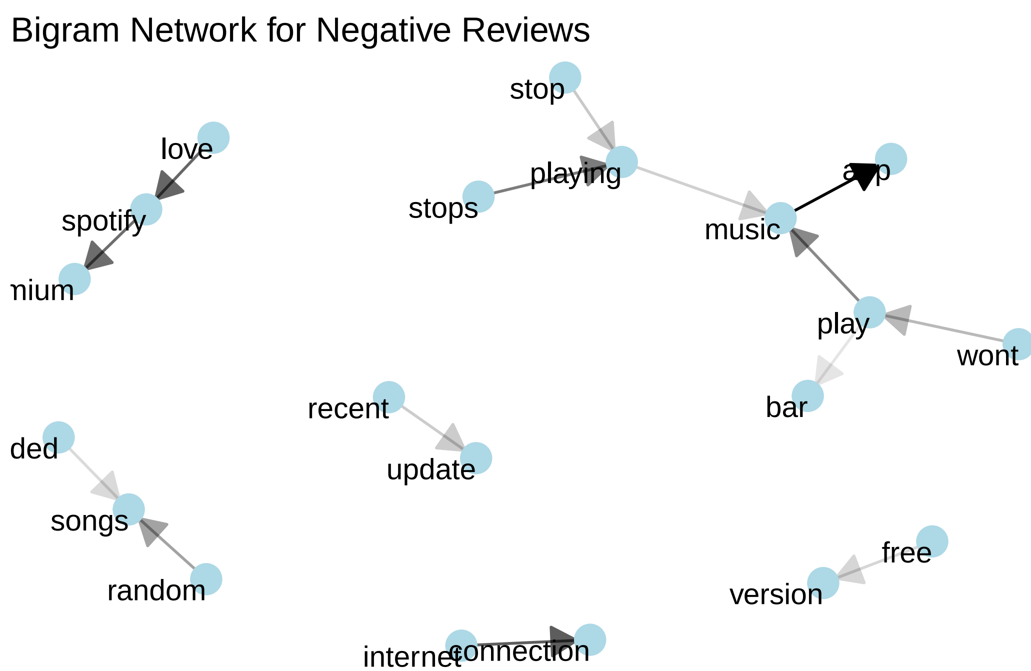
Create graphs visualizing the networks of bigrams, broken down by `label`. That is, make one graph of the network of bigrams for the positive reviews, and one graph of the network for the negative reviews.

```
spotify_bigrams_sep <- spotify_sample %>%
  unnest_tokens(bigram, Review, token = "ngrams", n = 2) %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word) %>%
```

```
  filter(!word2 %in% stop_words$word) %>%
  filter(!is.na(word1), !is.na(word2))

a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

# Positive
positive_bigrams <- spotify_bigrams_sep %>%
  filter(label == "POSITIVE") %>%
  count(word1, word2) %>%
  filter(n > 50) %>%
  graph_from_data_frame()

positive_plot <- ggraph(positive_bigrams, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.07, 'inches')) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
  ggtitle("Bigram Network for Positive Reviews") +
  theme_void()
positive_plot
```



```
# Negative
negative_bigrams <- spotify_bigrams_sep %>%
  filter(label == "NEGATIVE") %>%
  count(word1, word2) %>%
  filter(n > 50) %>%
  graph_from_data_frame()

negative_plot <- ggraph(negative_bigrams, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.07, 'inches')) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
  ggtitle("Bigram Network for Negative Reviews")+
  theme_void()
negative_plot
```



What patterns do you notice?

The arrows really help us interpret the bigram network. We can see phrases like "music app," "nice app," and "love spotify" in positive reviews. In negative reviews, people are talking about issues such as "stop playing," "internet connection," and "random songs." The network for negative reviews shows a greater number of connected words compared to positive reviews, suggesting that users discuss a wider range of concerns when expressing dissatisfaction.

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;

```
set.seed(123)
spotify_df <- tf_idf %>%
  mutate(label = factor(label)) %>%
  select(-word)

data_split <- initial_split(spotify_df, prop = 0.7, strata = label)
train_data <- training(data_split)
test_data <- testing(data_split)
cv_folds <- vfold_cv(train_data, v = 5, strata = label)
```

- Drop any columns with zero variance;

```
recipe <- recipe(label ~ ., data = train_data) %>%
  step_zv(all_predictors())

prep(recipe) %>% bake(train_data) %>% head()
```

```
# A tibble: 6 x 6
     id     n     tf   idf tf_idf label
  <int> <int>  <dbl> <dbl>  <dbl> <fct>
1    10     1 0.0345 2.35  0.0811 NEGATIVE
2    10     3 0.103  0.839 0.0868 NEGATIVE
3    10     1 0.0345 5.90  0.204  NEGATIVE
4    10     1 0.0345 7.41  0.255  NEGATIVE
5    10     1 0.0345 8.10  0.279  NEGATIVE
6    10     1 0.0345 5.49  0.189  NEGATIVE
```

- Fit a linear support vector machine using default values for any hyperparameters;

```
model <- svm_linear() %>%
  set_mode("classification") %>%
  set_engine("LiblineaR")

svm_workflow <- workflow() %>%
  add_recipe(recipe) %>%
```

```
  add_model(model)

svm_tune <- fit_resamples(
  svm_workflow,
  cv_folds,
  metrics = metric_set(accuracy))

final_fit <- fit(svm_workflow, data = train_data)
```

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

```
test_results <- final_fit %>%
  predict(new_data = test_data) %>%
  bind_cols(test_data) %>%
  metrics(truth = label, estimate = .pred_class) %>%
  filter(.metric == "accuracy")
test_results
```

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.702
```

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.