Module 3 Assignment 1

1.Load the packages tidytext and tidyverse

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.1.0 v dplyr 1.0.5  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(tidytext)

1. Read-in the review dataset for Airbnb listings in Boston in R (the dataset is available on Canvas). Change the name of the variable id to comment\_id.

#read in the data  
reviews <- read\_csv("reviews-boston.csv")  
  
reviews<-reviews %>%  
select(listing\_id, comment\_id=id,date,reviewer\_id,reviewer\_name, comments)

##bigrams 3. Create a tidytext dataset of guests’ reviews. Tokenize by bigrams.

(10 points)

reviews\_bigrams <- reviews %>%  
 unnest\_tokens(bigram, comments, token = "ngrams", n = 2)  
reviews\_bigrams

## # A tibble: 6,515,607 x 6  
## listing\_id comment\_id date reviewer\_id reviewer\_name bigram   
## <dbl> <dbl> <chr> <dbl> <chr> <chr>   
## 1 3781 37776825 7/10/2015 36059247 Greg the apartment   
## 2 3781 37776825 7/10/2015 36059247 Greg apartment was   
## 3 3781 37776825 7/10/2015 36059247 Greg was as   
## 4 3781 37776825 7/10/2015 36059247 Greg as advertised   
## 5 3781 37776825 7/10/2015 36059247 Greg advertised and   
## 6 3781 37776825 7/10/2015 36059247 Greg and frank   
## 7 3781 37776825 7/10/2015 36059247 Greg frank was   
## 8 3781 37776825 7/10/2015 36059247 Greg was incredibly   
## 9 3781 37776825 7/10/2015 36059247 Greg incredibly helpful  
## 10 3781 37776825 7/10/2015 36059247 Greg helpful through   
## # ... with 6,515,597 more rows

## Pre-processing and Word Frequesncy:

1. Filter stop-words, undesirable words, and words with less than 3 charachters (use separate () and filter()).

bigrams\_separated <- reviews\_bigrams %>%  
 separate(bigram, c("word1", "word2"), sep = " ")

(10 points)

undesirable\_words <- c("boston", "Boston", "stay", "host", "home", "house", "apartment")  
  
bigrams\_filtered <- reviews\_bigrams %>%  
 separate(bigram, c("word1", "word2"), sep = " ") %>%  
 filter(!word1 %in% stop\_words$word,  
 !word2 %in% stop\_words$word )%>%   
 filter(!word1 %in% undesirable\_words,  
 !word2 %in% undesirable\_words)%>%   
 filter(!nchar(word1) < 3,  
 !nchar(word2) < 3)

Find the most common bigrams in the comments (use count()).Explain the results.

(10 points)

The n-grams “highly recommend” and “walking distance” show up more than 7000 times, setting them in a class by themselves. It’s also worth nothing the “highly recommended” shows up more than 2000 times. Other ngrams are very positive and convey a feeling or thought about location or getting around the area. The famed Boston attraction called “freedom trail” shows up a lot in the reviews.

bigram\_counts <- bigrams\_filtered %>%   
 count(word1, word2, sort = TRUE)  
  
bigram\_counts

## # A tibble: 269,733 x 3  
## word1 word2 n  
## <chr> <chr> <int>  
## 1 highly recommend 7785  
## 2 walking distance 7465  
## 3 minute walk 3461  
## 4 public transportation 2858  
## 5 perfect location 2778  
## 6 short walk 2522  
## 7 recommend staying 2304  
## 8 easy access 2211  
## 9 highly recommended 2074  
## 10 freedom trail 1765  
## # ... with 269,723 more rows

Unite the words in the bigrams (use unite()).

(10 points)

bigrams\_united <- bigrams\_filtered %>%  
 unite(bigram, word1, word2, sep = " ")  
bigrams\_united

## # A tibble: 721,753 x 6  
## listing\_id comment\_id date reviewer\_id reviewer\_name bigram   
## <dbl> <dbl> <chr> <dbl> <chr> <chr>   
## 1 3781 37776825 7/10/2015 36059247 Greg incredibly helpful  
## 2 3781 37776825 7/10/2015 36059247 Greg entire process   
## 3 3781 41842494 8/9/2015 10459388 Tai maverick square   
## 4 3781 41842494 8/9/2015 10459388 Tai min train   
## 5 3781 41842494 8/9/2015 10459388 Tai train ride   
## 6 3781 41842494 8/9/2015 10459388 Tai street parking   
## 7 3781 41842494 8/9/2015 10459388 Tai private spot   
## 8 3781 41842494 8/9/2015 10459388 Tai spot frank   
## 9 3781 45282151 9/1/2015 12264652 Damien nice view   
## 10 3781 45282151 9/1/2015 12264652 Damien quiet relaxing   
## # ... with 721,743 more rows

## TF-IDF

1. TF-IDF: Calculate the tf-idf values of bigrams to identify unique bigrams for each comment\_id. Explain the results.

(15 points)

The bigrams that show up when sorted by descending tf-idf show pairs of words that are unique and which review they belong to. These words are what sets the review apart and make it special among all the other reviews. The tf-idf analysis removes the ordinary words and highlights the special ones. “received warm”, “extended amount”, and “included parking” imply uniqueness and specialness to their respective reviews.

bigram\_tf\_idf <- bigrams\_united %>%  
 count(listing\_id, bigram) %>%  
 bind\_tf\_idf(bigram, listing\_id, n) %>%  
 arrange(desc(tf\_idf))  
bigram\_tf\_idf

## # A tibble: 618,081 x 6  
## listing\_id bigram n tf idf tf\_idf  
## <dbl> <chr> <int> <dbl> <dbl> <dbl>  
## 1 13642024 received warm 1 1 6.94 6.94  
## 2 16983905 extended amount 1 1 6.94 6.94  
## 3 8097453 included parking 1 1 4.93 4.93  
## 4 15267276 diverse neighborhood 1 1 4.27 4.27  
## 5 12023820 short weekend 1 1 4.11 4.11  
## 6 4315439 price payed 1 0.5 7.64 3.82  
## 7 7592616 neighborhood bryan 1 0.5 7.64 3.82  
## 8 7592616 snacks bryan 1 0.5 7.64 3.82  
## 9 9166393 wise advices 1 0.5 7.64 3.82  
## 10 9349646 caring couple 1 0.5 7.64 3.82  
## # ... with 618,071 more rows

## Sentiment analysis - bigrams:

1. Let’s use the AFINN lexicon for sentiment analysis

AFINN <- get\_sentiments("afinn")

Pick the common words that negate the subsequent word like: “not”, “no”, “never”, “without”, etc.

(15 points)

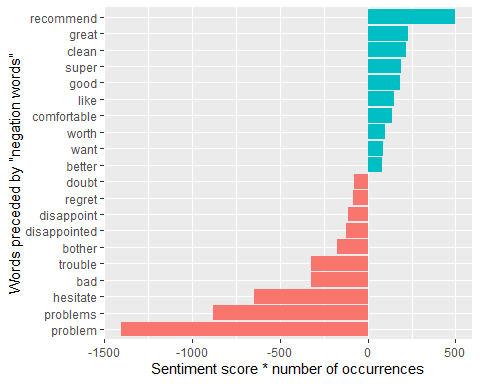
negation\_words <- c("not", "no", "never", "without")  
negated\_words <- bigrams\_separated %>%  
 filter(word1 %in% negation\_words) %>%  
 inner\_join(AFINN, by = c(word2 = "word")) %>%  
 count(word1, word2, value, sort = TRUE)  
negated\_words

## # A tibble: 451 x 4  
## word1 word2 value n  
## <chr> <chr> <dbl> <int>  
## 1 no problem -2 703  
## 2 no problems -2 439  
## 3 not hesitate -2 323  
## 4 not recommend 2 250  
## 5 no trouble -2 160  
## 6 not clean 2 109  
## 7 not bad -3 108  
## 8 not want 1 90  
## 9 not bother -2 87  
## 10 not great 3 77  
## # ... with 441 more rows

1. Visualize and explain the results. What are the most common sentiment-associated word from the lexicon AFINN to follow a negation word? Words that most frequently followed by negation words are the largest causes of mis-identifications.

“recommend” is the word most frequently mis-identified leading to a total score of 500. This is understandable with having the words “no” or “not” in front of it. On the positive side, the words “problem” and “problems” are the negative words that are frequently mis-identified, with scores of over 1400 and almost 900. Again, the use of the phrase “no problem” doesn’t seem like a big deal, but this is a case where two negatives make a positive.

negated\_words %>%  
 mutate(contribution = n \* value) %>%  
 arrange(desc(abs(contribution))) %>%  
 head(20) %>%  
 mutate(word2 = reorder(word2, contribution)) %>%  
 ggplot(aes(word2, n \* value, fill = n \* value > 0)) +  
 geom\_col(show.legend = FALSE) +  
 xlab("Words preceded by \"negation words\"") +  
 ylab("Sentiment score \* number of occurrences") +  
 coord\_flip()



1. Load the package igraph:

# install.packages("igraph")  
library(igraph)

# install.packages("widyr")  
# install.packages("ggraph")  
library(widyr)  
library(dplyr)

1. Use the output “bigram\_counts” from step 4 and build a network of common bigrams [filter for only relatively common combinations (n>500) – use lines instead of directed arrows between nodes (graph\_from\_data\_frame (directed = FALSE))].

(15 points)

# filter for only relatively common combinations  
set.seed(1234)  
bigram\_graph <- bigram\_counts %>%  
 filter(n >= 500) %>%  
 graph\_from\_data\_frame(directed = FALSE)

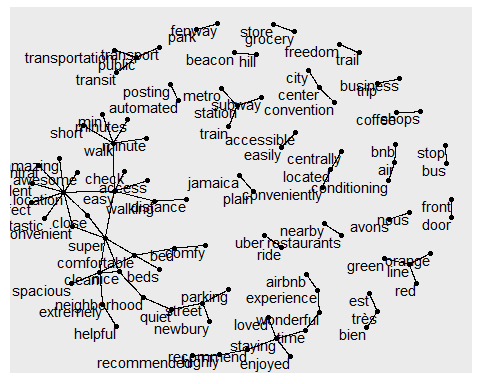
1. Load the package ggraph

library(ggraph)

## Warning: package 'ggraph' was built under R version 4.0.5

1. Visualize the graph - Use the Fruchterman-Reingold to visualize the nodes and ties (“fr”). Explain the results.

set.seed(2017)  
  
ggraph(bigram\_graph, layout = "fr") +  
 geom\_edge\_link() +  
 geom\_node\_point() +  
 geom\_node\_text(aes(label = name), vjust = 1, hjust = 1)

 From this directed graph we can see what pairs of words are commonly used together. it’s filled with lots of tourist attractions (“Fenway Park”, “Beacon Hill”, “Freedom Trail”), and some phrases that always go together, like “grocery stores”, “coffee shops”, “front door”, and “business trip”. The biggest network deals with beds, neighborhoods, and walks, which are all described with similar adjectives like comfortable, close, super, and clean.

1. Apply some polishing operations to make a better looking graph.

* add the edge\_alpha aesthetic to the link layer to make links transparent based on how common or rare the bigram is
* add the edge\_width aesthetic to the link layer to show the weight of the ties between bigrams
* add edge\_colour = “cyan4” to the link layer
* tinker with the options to the node layer to make the nodes smaller (size=1 with diffult black color)
* add a theme that’s useful for plotting networks, theme\_void()

(15 points)

set.seed(2016)  
  
ggraph(bigram\_graph, layout = "fr") +  
 geom\_edge\_link(aes(edge\_alpha = n, edge\_width = n), edge\_colour = "cyan4") +  
 geom\_node\_point(size = 1, colour = "black") +  
 geom\_node\_text(aes(label = name), size=3, repel=TRUE, point.padding=unit(0.2,"lines"), vjust = 1, hjust = 1)+  
 theme\_void()

