Analyzing Phrases with tidytext

As always, we set the working directory and then load the packages we want.

library(tidyverse)  
library(tidytext)  
library(okcupiddata)  
library(scales) # scale functions for visualization  
library(igraph)  
library(ggraph)

We continue the analysis of the OKCupid essays, and just to add some further interests we compare responses of smokers and non-smokers.

For clarity, let’s break the process down into steps. First, subset the okcupid profiles to include just the essay responses and the user’s response about smoking. The data\_frame function creates a tibble. In creating the tibble, we rename essay0 as text – not necessary, but follows the example in Silge and Robinson.

Then we create a new binary variable, smoker that identifies non-smokers as “no” and everyone else as “yes”, dropping the original smokes variable (notice the command select(-smokes), which selects all columns *except for* smokes.

tidy\_okcupid <- select\_(profiles,"essay0","smokes")   
  
tidy\_okcupid <- data\_frame(smokes=profiles$smokes,text=profiles$essay0)  
tidy\_okcupid <- tidy\_okcupid %>%  
 mutate(smoker = ifelse(smokes=="no","no","yes")) %>%   
 select(-smokes)

The next few lines of code *remove* all NA rows, because the unnest\_tokens function requires complete cases. The last line in this chunk unnests 2-word phrases from the . You may also want to experiment with longer phrases.

tidy\_smoker <- tidy\_okcupid %>%  
 select(smoker, text) %>%  
 na.omit()  
  
smoker\_bigrams <- unnest\_tokens(tidy\_smoker, bigram, text, token="ngrams", n=2)  
  
smoker\_bigrams

## # A tibble: 1,226,066 x 2  
## smoker bigram   
## <chr> <chr>   
## 1 no i am   
## 2 no am a   
## 3 no a chef   
## 4 no chef this   
## 5 no this is   
## 6 no is what   
## 7 no what that   
## 8 no that means  
## 9 no means 1   
## 10 no 1 i   
## # ... with 1,226,056 more rows

At this point, the tibble smoker\_bigrams consists of more than 1,226,000 pairs of adjacent words. The next code chunk counts and sorts the pairs to find the most common ones. After that, it separates the pairs into word1 and word2 to search for and remove stopwords, and then unites the pairs that remain, finally recounting and sorting the shorter list of pairs.

Consult Silge & Robinson to learn how to customize the list of stopwords.

smoker\_bigrams %>%  
 count(bigram, sort = TRUE)

## # A tibble: 323,132 x 2  
## bigram n  
## <chr> <int>  
## 1 i am 14243  
## 2 i'm a 6571  
## 3 i love 6445  
## 4 in the 5275  
## 5 i like 4480  
## 6 i have 4346  
## 7 am a 4215  
## 8 and i 3382  
## 9 san francisco 3269  
## 10 like to 3128  
## # ... with 323,122 more rows

bigrams\_separated <- smoker\_bigrams %>%  
 separate(bigram, c("word1", "word2"), sep = " ")  
   
bigrams\_filtered <- bigrams\_separated %>%  
 filter(!word1 %in% stop\_words$word) %>%  
 filter(!word2 %in% stop\_words$word)  
   
# new bigram counts:  
bigram\_counts <- bigrams\_filtered %>%   
 count(word1, word2, smoker, sort = TRUE)  
   
  
bigram\_counts

## # A tibble: 105,832 x 4  
## word1 word2 smoker n  
## <chr> <chr> <chr> <int>  
## 1 san francisco no 2669  
## 2 east coast no 750  
## 3 san francisco yes 600  
## 4 fun loving no 537  
## 5 recently moved no 449  
## 6 east bay no 254  
## 7 online dating no 229  
## 8 enjoy life no 214  
## 9 grad school no 208  
## 10 san diego no 187  
## # ... with 105,822 more rows

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

## # A tibble: 159,694 x 2  
## smoker bigram   
## <chr> <chr>   
## 1 no means 1   
## 2 no workaholic 2   
## 3 no writing public   
## 4 no public text   
## 5 no online dating   
## 6 no dating site   
## 7 no site makes   
## 8 no pleasantly uncomfortable  
## 9 no school hey   
## 10 no australian living   
## # ... with 159,684 more rows

As noted in class and in the readings, a simple frequency count can be misleading. The next code chunk weights term frequency by the inverse frequency within the documents (smokers vs non-smokers.

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

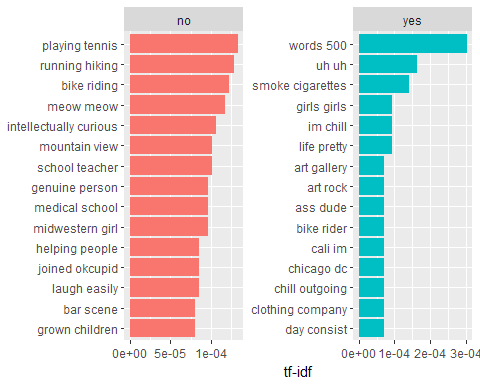
## # A tibble: 105,832 x 6  
## smoker bigram n tf idf tf\_idf  
## <chr> <chr> <int> <dbl> <dbl> <dbl>  
## 1 yes words 500 13 0.000436 0.693 0.000302  
## 2 yes uh uh 7 0.000235 0.693 0.000163  
## 3 yes smoke cigarettes 6 0.000201 0.693 0.000140  
## 4 no playing tennis 25 0.000192 0.693 0.000133  
## 5 no running hiking 24 0.000185 0.693 0.000128  
## 6 no bike riding 23 0.000177 0.693 0.000123  
## 7 no meow meow 22 0.000169 0.693 0.000117  
## 8 no intellectually curious 20 0.000154 0.693 0.000107  
## 9 no mountain view 19 0.000146 0.693 0.000101  
## 10 no school teacher 19 0.000146 0.693 0.000101  
## # ... with 105,822 more rows

bigrams\_separated %>%  
 filter(word1 == "not") %>%  
 count(word1, word2, sort = TRUE)

## # A tibble: 995 x 3  
## word1 word2 n  
## <chr> <chr> <int>  
## 1 not a 365  
## 2 not to 227  
## 3 not really 213  
## 4 not sure 196  
## 5 not looking 165  
## 6 not the 146  
## 7 not so 111  
## 8 not in 97  
## 9 not very 96  
## 10 not too 84  
## # ... with 985 more rows

plot\_pairs <- bigram\_tf\_idf %>%  
 arrange(desc(tf\_idf)) %>%  
 mutate(pair = factor(bigram, levels = rev(unique(bigram))))  
  
plot\_pairs %>%   
 group\_by(smoker) %>%   
 top\_n(15) %>%   
 ungroup %>%  
 ggplot(aes(pair, tf\_idf, fill = smoker)) +  
 geom\_col(show.legend = FALSE) +  
 labs(x = NULL, y = "tf-idf") +  
 facet\_wrap(~smoker, ncol = 2, scales = "free") +  
 coord\_flip()

## Selecting by pair



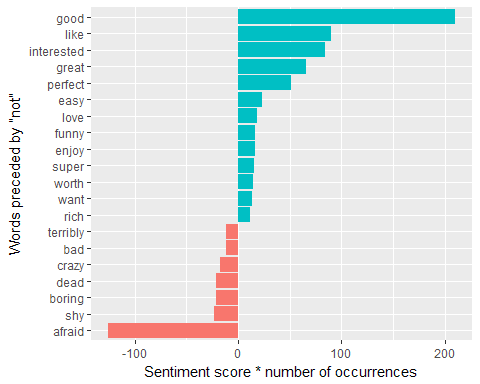
# Sentiment Analysis

DataCamp introduced you to Sentiment Analysis. This next chunk follows the Silge & Robinson example. We specifically look for word pairs beginning that include negation (like “not”), since that can be a common style.

AFINN <- get\_sentiments("afinn")  
   
 not\_words <- bigrams\_separated %>%  
 filter(word1 == "not") %>%  
 inner\_join(AFINN, by = c(word2 = "word")) %>%  
 count(word2, score, sort = TRUE) %>%  
 ungroup()  
   
 not\_words

## # A tibble: 168 x 3  
## word2 score n  
## <chr> <int> <int>  
## 1 good 3 70  
## 2 afraid -2 63  
## 3 like 2 45  
## 4 interested 2 42  
## 5 easy 1 23  
## 6 shy -1 23  
## 7 great 3 22  
## 8 perfect 3 17  
## 9 want 1 13  
## 10 crazy -2 9  
## # ... with 158 more rows

not\_words %>%  
 mutate(contribution = n \* score) %>%  
 arrange(desc(abs(contribution))) %>%  
 head(20) %>%  
 mutate(word2 = reorder(word2, contribution)) %>%  
 ggplot(aes(word2, n \* score, fill = n \* score > 0)) +  
 geom\_col(show.legend = FALSE) +  
 xlab("Words preceded by \"not\"") +  
 ylab("Sentiment score \* number of occurrences") +  
 coord\_flip()



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, score, sort = TRUE) %>%  
 ungroup()

At this point we begin to make network graphs, first showing the most common word combinations for *non-smokers. You could then repeat and modify the chunk for smokers*.

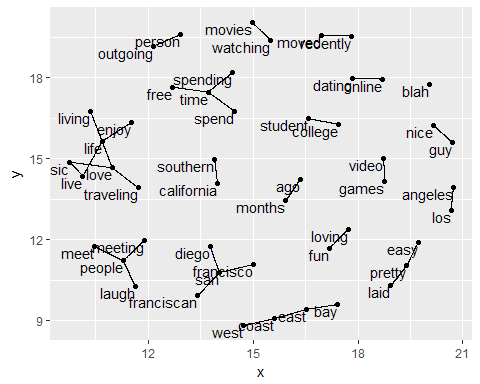
# original counts  
 bigram\_counts

## # A tibble: 105,832 x 4  
## word1 word2 smoker n  
## <chr> <chr> <chr> <int>  
## 1 san francisco no 2669  
## 2 east coast no 750  
## 3 san francisco yes 600  
## 4 fun loving no 537  
## 5 recently moved no 449  
## 6 east bay no 254  
## 7 online dating no 229  
## 8 enjoy life no 214  
## 9 grad school no 208  
## 10 san diego no 187  
## # ... with 105,822 more rows

# filter for only relatively common combinations  
# might want to experiment with values other than 20 in filter line  
 bigram\_graph <- bigram\_counts %>%  
 group\_by(smoker) %>%  
 filter(n > 20) %>%  
 filter(smoker=="yes") %>%  
 graph\_from\_data\_frame()  
   
 bigram\_graph

## IGRAPH 77c3155 DN-- 49 33 --   
## + attr: name (v/c), smoker (e/c), n (e/n)  
## + edges from 77c3155 (vertex names):  
## [1] san ->francisco recently->moved fun ->loving   
## [4] east ->coast love ->music east ->bay   
## [7] online ->dating san ->diego months ->ago   
## [10] blah ->blah people ->laugh pretty ->laid   
## [13] meet ->people nice ->guy pretty ->easy   
## [16] video ->games enjoy ->life live ->life   
## [19] free ->time west ->coast love ->life   
## [22] san ->franciscan southern->california love ->traveling   
## + ... omitted several edges

set.seed(2018)  
   
 ggraph(bigram\_graph, layout = "fr") +  
 geom\_edge\_link() +  
 geom\_node\_point() +  
 geom\_node\_text(aes(label = name), vjust = 1, hjust = 1)



Finally, as in Chapter 4 of Silge and Robinson (which was not assigned), here’s an alternative and more attractive network graph. This will use the same subset of participants as the one created above.

set.seed(2018)  
   
 a <- grid::arrow(type = "closed", length = unit(.05, "inches"))  
   
 ggraph(bigram\_graph, layout = "fr") +  
 geom\_edge\_link(aes(edge\_alpha = n), show.legend = FALSE,  
 arrow = a, end\_cap = circle(.05, 'inches')) +  
 geom\_node\_point(color = "lightblue", size = 5) +  
 geom\_node\_text(aes(label = name), vjust = 1, hjust = 1) +  
 theme\_void()

