140SL Text Analysis

Josh Kong, Jason Chhay, Atif Farook, Raymond Astorga, Richard Garcia

12/8/2020

Loading libraries and dataset

```r
library(tidyverse)
```

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

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(tidytext)
library(tidylo)
listings <- read_csv("listings_cleaned_2clus.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]

##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   id = col_double(),
##   name = col_character(),
##   description = col_character(),
##   neighborhood_overview = col_character(),
##   host_id = col_double(),
##   price = col_double(),
##   value = col_character()
## )
```

```r
listings <- listings[,-1]
listings$value <- factor(listings$value)
```

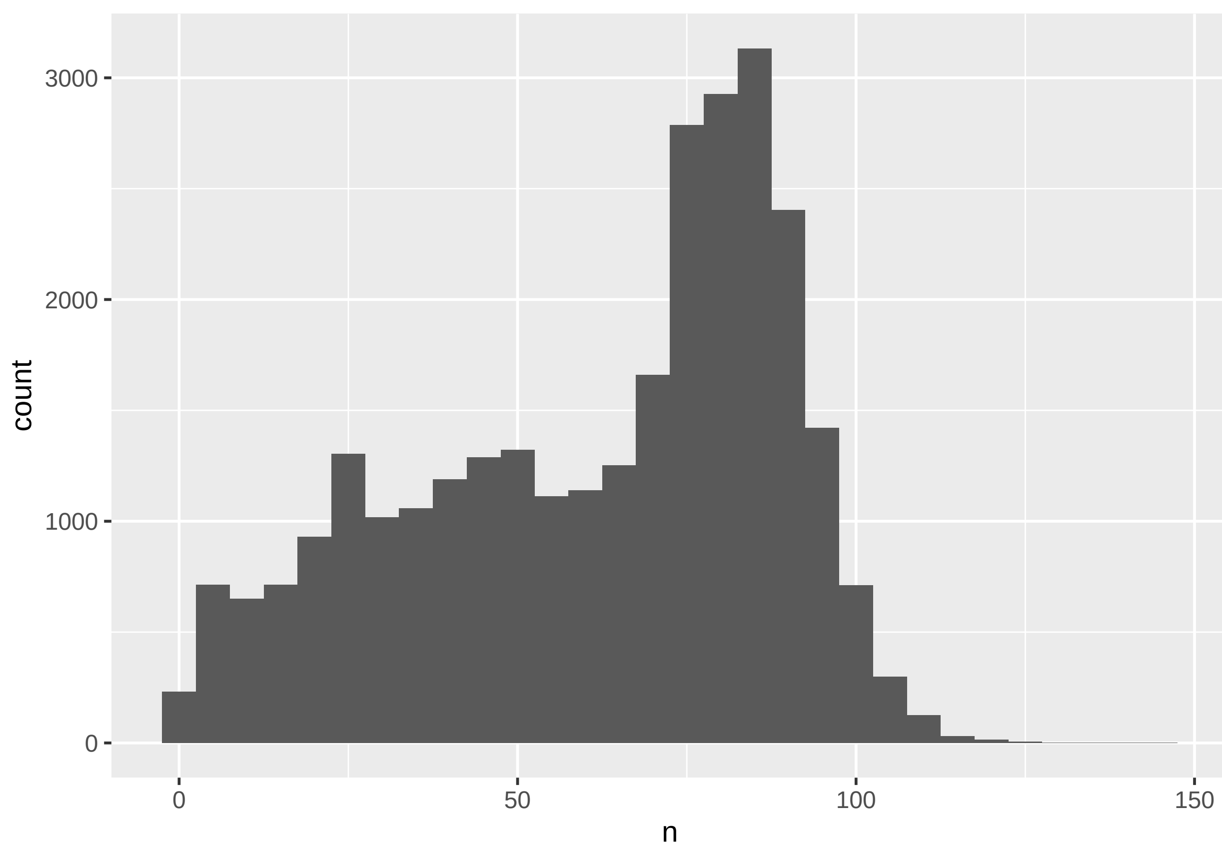
Low is 138 dollars or under. High is 139 dollars or higher

Exploring the dataset

```r
#unnesting the description column and filtering out words with special symbols.
listings_unnested <-  listings %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words, by = "word") %>%
  filter(!str_detect(word,"[^\x01-\x7F]"))

#looking at the distribution of the length of the descriptions
listings_unnested %>%
  count(id, sort = TRUE) %>%
  ggplot(aes(n)) +
  geom_histogram()
```

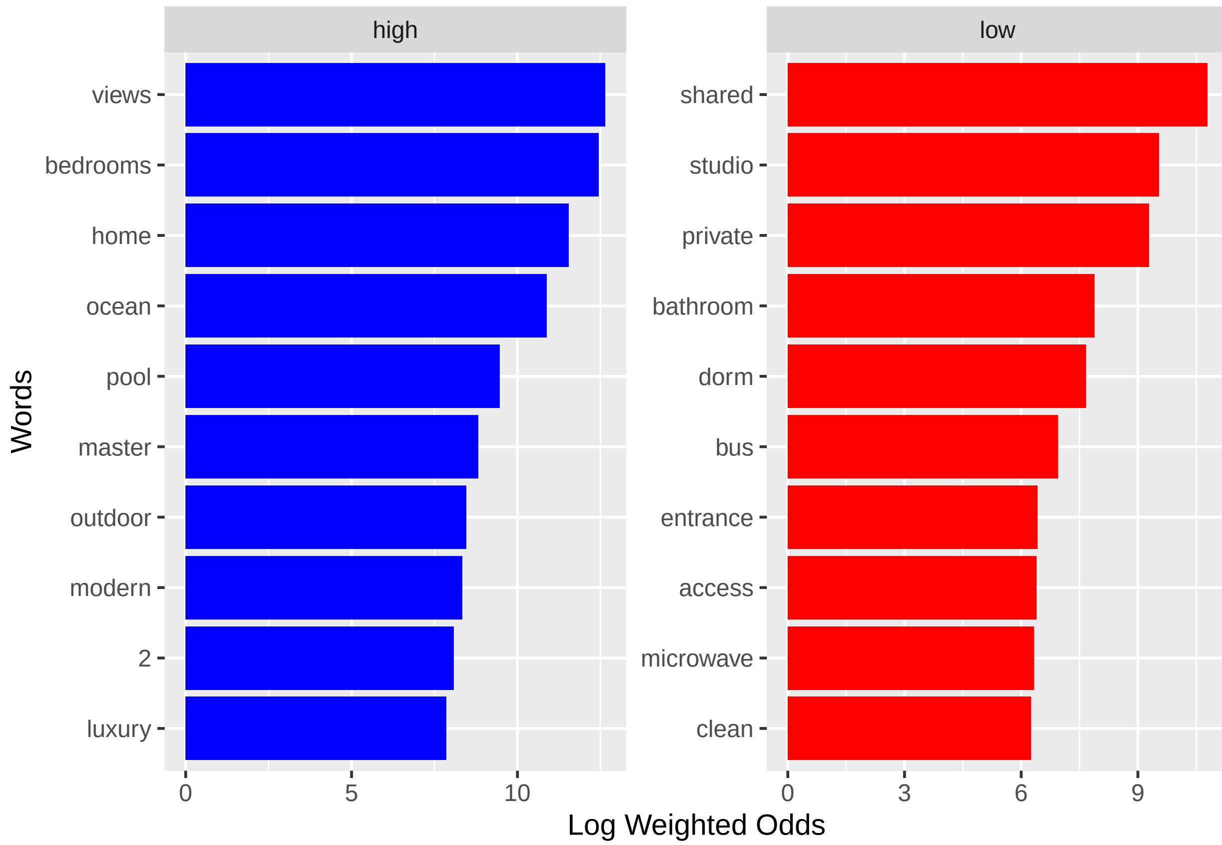
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Taking a look at the log weighted odds of which words belong in the high value air bnb descriptions, and which words belong in the low value air bnb descriptions.

```r
listings_lo <- listings_unnested %>%
  count(value, word) %>%
  bind_log_odds(value, word,n) %>%
  arrange(-log_odds_weighted)
```

```r
listings_lo %>%
  group_by(value) %>%
  slice_max(log_odds_weighted, n = 10) %>%
  ungroup() %>%
  mutate(word = reorder(word, log_odds_weighted)) %>%
  ggplot(aes(word, log_odds_weighted, fill = value)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~value, scales = "free") +
  scale_fill_manual(values = c("blue","red"))+
  coord_flip() +
  labs(x = "Words", y = "Log Weighted Odds")
```



Building the Model

Splitting the data into training and testing sets

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.2 --

## v broom     0.7.2      v recipes   0.1.15
## v dials     0.0.9      v rsample   0.0.8
```

```
## v infer     0.5.3      v tune      0.1.2
## v modeldata 0.1.0      v workflows 0.2.1
## v parsnip   0.1.4      v yardstick 0.0.7

## -- 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()
```

```
set.seed(123)
listings_split <- initial_split(listings, strata = value)
listings_train <- training(listings_split)
listings_test <- testing(listings_split)
```

Preparing the data for model building

```
library(themis)
```

```
## Registered S3 methods overwritten by 'themis':
##   method                  from
##   bake.step_downsample    recipes
##   bake.step_upsample      recipes
##   prep.step_downsample    recipes
##   prep.step_upsample      recipes
##   tidy.step_downsample    recipes
##   tidy.step_upsample      recipes
##   tunable.step_downsample recipes
##   tunable.step_upsample   recipes

##
## Attaching package: 'themis'

## The following objects are masked from 'package:recipes':
##
##     step_downsample, step_upsample
```

```
library(textrecipes)
listings_rec <- recipe(value ~ description, data = listings_train) %>%
  step_tokenize(description) %>%      #unnests the description
  step_stopwords(description) %>%     #getting rid of stopwords such as "a" and "the"
  step_tokenfilter(description, max_tokens = 500) %>%  #filter for top 500 words that were present
  step_tfidf(description) %>%   #converting the words into variables. value is given based on frequency
  step_normalize(all_predictors())   #Normalizing all the variables
```

Creating model specifications

```
lasso_spec <- logistic_reg(penalty = tune(), mixture = 1) %>%  #going to tune penalty
  set_engine("glmnet")

lasso_wf <- workflow() %>%          #adding model and recipe to a workflow
  add_recipe(listings_rec) %>%
  add_model(lasso_spec)
```

Tuning the model

```
lambda_grid <- grid_regular(penalty(), levels = 20)  #a grid with 20 different lambda values to try

set.seed(123)
review_folds <- vfold_cv(listings_train, strata = value, v = 5) #5 fold cross validation to test differ

doParallel::registerDoParallel()  #speeds up the process
set.seed(5)
lasso_grid <- tune_grid(
  lasso_wf,                         #going to tune the lasso model
  resamples = review_folds,       #resampling with the cv folds made earlier
  grid = lambda_grid,             #using the 20 different lambda values
  metrics = metric_set(accuracy) #metric of success is going to be accuracy of predictions
)
```

```
##
## Attaching package: 'rlang'

## The following objects are masked from 'package:purrr':
##
##     %@%, as_function, flatten, flatten_chr, flatten_dbl, flatten_int,
##     flatten_lgl, flatten_raw, invoke, list_along, modify, prepend,
##     splice

##
## Attaching package: 'vctrs'

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

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

## Loading required package: Matrix

##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.0-2
```

```
best_tune <- lasso_grid %>%       #selecting best penalty value based on accuracy
  select_best("accuracy")

final_model <- finalize_workflow(lasso_wf, best_tune)   #finalizing the model
```

Fitting the model

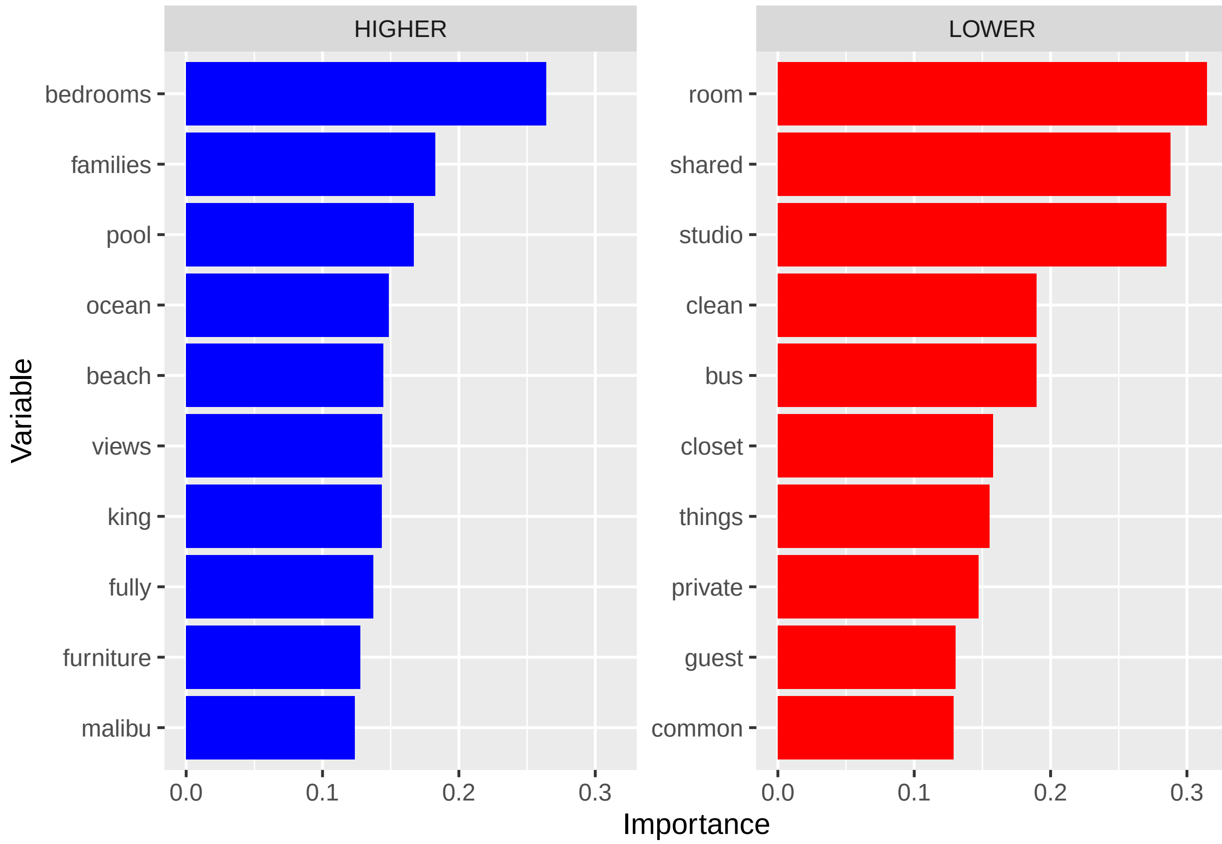
```
lasso_fit <- final_model %>%    #fitting the training data to the model
  fit(listings_train)

#observing which words in the description lead to higher prices
library(vip)
```

```
##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##     vi
```

```
lasso_fit %>%
  pull_workflow_fit() %>%
  vi(lambda = best_tune$penalty) %>%
  group_by(Sign) %>%
  top_n(10, wt = abs(Importance)) %>%
  ungroup() %>%
  mutate(Importance = abs(Importance),
         Variable = str_remove(Variable, "tfidf_description_"),
         Variable = fct_reorder(Variable, Importance),
         effect = ifelse(Sign == "NEG", "HIGHER", "LOWER")) %>%
  ggplot(aes(x = Importance, y = Variable, fill = effect)) +
  geom_col(show.legend = FALSE) +
  scale_fill_manual(values = c("blue","red"))+
  facet_wrap(~effect, scales = "free_y")
```



Testing the model

```
lasso_test <- final_model %>%   #fitting the final model to the testing data
  fit(listings_test)
lasso_pred <- predict(lasso_test, listings_test)

#confidence matrix of actual values vs predicted values
conf_mat <- table(prediction = lasso_pred$.pred_class, actual = listings_test$value); conf_mat
```

```
##             actual
## prediction high  low
##       high 1972  546
##       low   939 4175
```

```
#accuracy of the predictions
(conf_mat[1,1] + conf_mat[2,2]) / sum(conf_mat)
```

```
## [1] 0.8054245
```

Getting a 80.5% accuracy rate.