Cleaning and preprocessing

Alan Nurcahyo

10/27/2020

Preprocessing

```
library(tidyverse)
```

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

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     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()
```

Load and review dataset

There are 423165 obs and 10 column in dataset. [1] "id": Unique Id for dish [2] "name": Name of dish [3] "description": [4] "menus_appeared": indicate how many menu list the dish in it [5] "times_appeared": indicate how many times dish appeared during first time appeared and last time appeared [6] "first_appeared": indicate year in which the dish first time seen [7] "last_appeared": indicate year in which the dish last time seen [8] "lowest_price": indicate the lowest price of dish ever recorded [9] "highest_price": indicate the highest price of dish ever recorded [10] "seafood_yn": indicate whether the dish is seafood or not a seafood. NA indicate that the dish are not labeled.

```
#load dataset
df <- read_csv("/Users/alannurcahyo/Documents/Documents/AU/fall2020/DATA/Seafood_Dishes.csv")
```

```
##
## -- Column specification --------------------------------------------------------------------------------
## cols(
##   id = col_double(),
##   name = col_character(),
##   description = col_logical(),
##   menus_appeared = col_double(),
##   times_appeared = col_double(),
##   first_appeared = col_double(),
```

```
##   last_appeared = col_double(),
##   lowest_price = col_double(),
##   highest_price = col_double(),
##   seafood_yn = col_double()
## )

## Warning: 1 parsing failure.
##  row        col expected actual
## 1213 seafood_yn a double     o0 '/Users/alannurcahyo/Documents/Documents/AU/fall2020/DATA/Seafood_Dis
```

```
## column names
names(df)
```

```
##  [1] "id"             "name"           "description"    "menus_appeared"
##  [5] "times_appeared" "first_appeared" "last_appeared"  "lowest_price"
##  [9] "highest_price"  "seafood_yn"
```

```
##check label
dim(df)
```

```
## [1] 423165     10
```

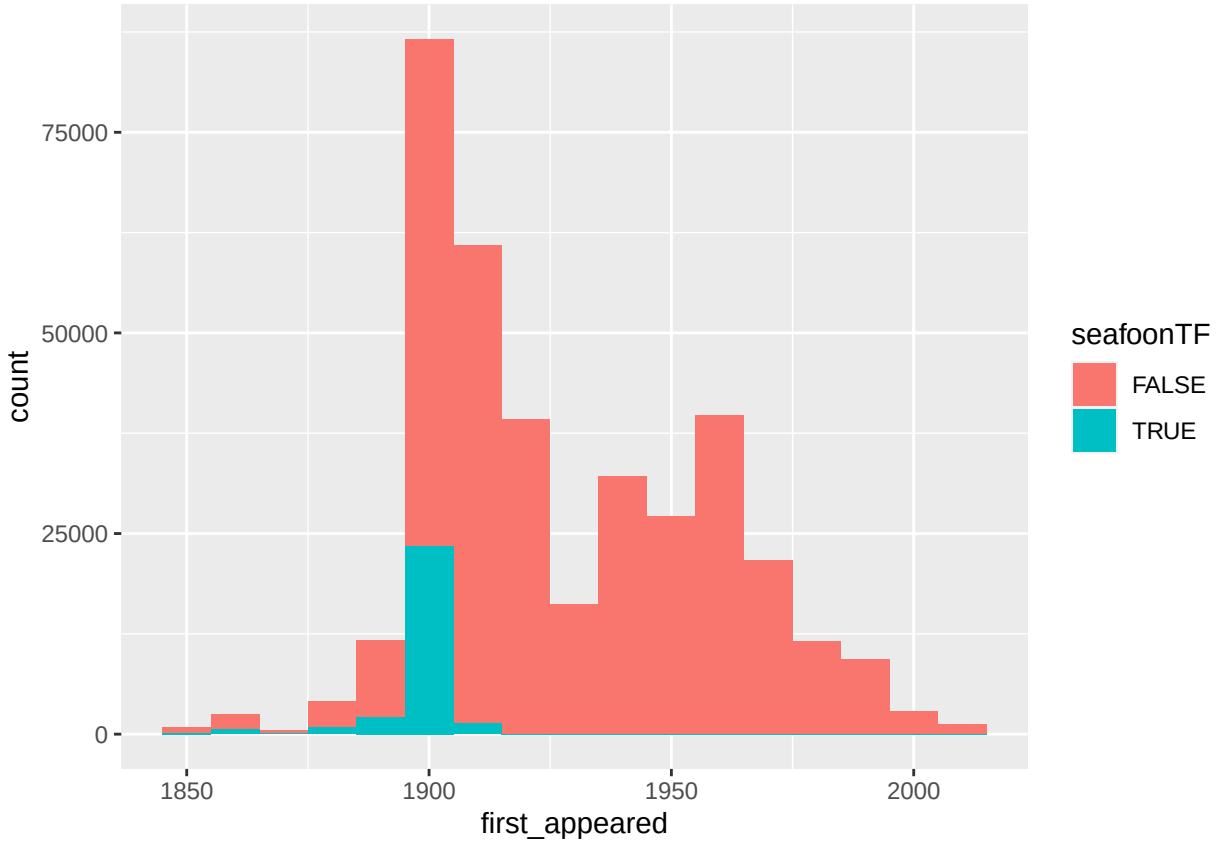
```
unique(df$seafood_yn)
```

```
## [1] NA  0  1
```

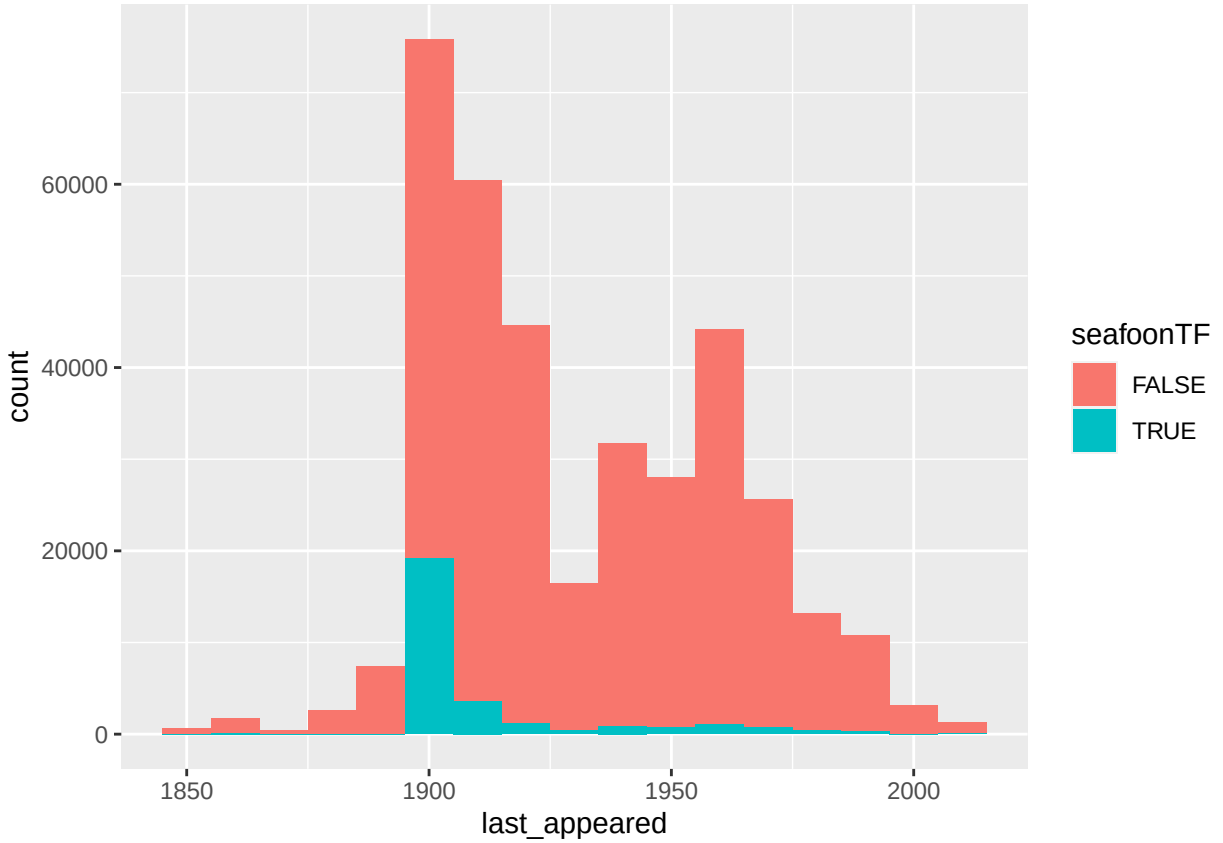
Distribution of labeled and non labeled by Year

the histogram below shows that the data lacks of observation from later year (after 1908) we can see that there are 246,640, only 58 of the are labeled.

```
####### Histogram for evaluation ##########
#Histogram of first time the menu appeared
# It appears that the labeled data lack of observation from later year
df %>%
  filter(first_appeared != 0,
         first_appeared != 1,
         first_appeared != 2928) %>%
  mutate( seafoonTF = !is.na(seafood_yn)) %>%
  ggplot(aes(first_appeared, fill = seafoonTF)) +
  geom_histogram(binwidth = 10)
```




```
#Histogram of last time the menu appeared
# It appears that the labeled data has a sufficient range of observation
df %>%
  filter(last_appeared != 0,
         last_appeared != 1,
         last_appeared != 2928) %>%
  mutate( seafoonTF = !is.na(seafood_yn)) %>%
  ggplot(aes(last_appeared, fill = seafoonTF)) +
  geom_histogram(binwidth = 10)
```




```
# comparison between labeled and unlabeled data by first appeared
# It appears that labeled data lack of observation after 1908
count.first.appeared <- df %>%
  filter(first_appeared != 0,
         first_appeared != 1,
         first_appeared != 2928) %>%
  mutate( seafoodTF = !is.na(seafood_yn)) %>%
  group_by(first_appeared, seafoodTF) %>%
  count() %>%
  spread(seafoodTF, n)

# number of observation after 1097
sum(count.first.appeared$`FALSE`[count.first.appeared$first_appeared>1907])
```

```
## [1] 246640
```

```
# number of labeled observation after 1097
sum(count.first.appeared$`TRUE`[count.first.appeared$first_appeared>1907], na.rm = TRUE)
```

```
## [1] 58
```

We need to generate new sample so that dataset has a vocabulary from later years. To do that, we take one percent of oberservation after 1908 to be labeled. To make sure we have good additional sample we take # observation based on it's proportion to dataset. Therefore, if we have more dishes in 1988 than 1990, we

take more sample from 1988 than 1990. This approach also making sure that we have at least one sample from all years. We have 2396 new observations to be labeled in the next step.

Finally, we label it 1 for seafood or 0 for non seafood, then we bind new labeled sample to dataset.

```
####### Generate new sample ##########

# let's take one percent of oberservation after 1908 to be labeled and round it
# per year we need following observation
n_new_sample <- count.first.appeared %>%
  mutate(n_sample = round(`FALSE`/100,0)) %>%
  select(first_appeared,n_sample) %>%
  filter(first_appeared >1907,
         n_sample >0)
#
# # ## we take n number of observation from unlabeled dataset by following the number we need on n_new_
new_sample <- vector(mode = "list", length = nrow(n_new_sample))
for (i in seq_along(new_sample)) {
  new_sample[[i]] <- df.unlabeled %>%
    filter(first_appeared == n_new_sample$first_appeared[[i]]) %>%
    sample_n(n_new_sample$n_sample[[i]])
}
new_sample <- do.call(rbind, new_sample)

# write it to csv
write_csv(new_sample,"new_sample.csv")
# # label it manually, then load the labeled new sample into R
new_sample_labeled <- read_csv("new_sample_labeled.csv")
dim(new_sample_labeled)
new_sample_labeled <- new_sample_labeled %>%
  filter(seafood_yn %in% c(0,1))

df <- df %>%
  anti_join(new_sample_labeled, by = "id")
df <- rbind(df, new_sample_labeled)
```

Translate

Name variable in the dataset is written in namy languages, including English, Germany, and French. This difference in language used causes several problems. firstly, it is difficult for cleaning procedure such as removing stop words and stemming. Secondly, the same dish will be seen as different dish is it's written in other language. Therefore, we will translate name variable in dataset using Google Translate API.

Using Google API, we make loop function that translate dish name one by one and save it into a new object. Next, we use 'cbind' to merge it with dataset as new variable nameTranslated.

```
####################### use google API ##########################
## the output file is saved in .csv because it is costly and time consuming to run it everytime
gl_auth("seafood-translation-f329aa685c7f.json")
df.name.translate <- vector(mode = "list", length = nrow(df))

for (i in seq_along(df.name.translate)) {
     df.name.translate[[i]] <- gl_translate(df$name[i], target = "en")
}
```

```
df.name.translate.bind <- do.call(rbind, df.name.translate)
df.name.translate.bind <- df.name.translate.bind %>%
  distinct()
write_csv(df.name.translate.bind, " translationgoogleall.csv")
df <- left_join(df, df.name.translate.bind)
```

Once we have all name in english, we can evaluate whether our our train sample has a proper vocabulary compared to the rest of dataset(unlabeled).

To do that, first we separate labeled and unlabeled dataset. Then we make train and test from labeled dataset. We will do this process again after we add more observation to enrich vocabulary in our dataset.

```
unlabeled <- df %>%
  filter(is.na(seafood_yn))
labeled <- df %>%
  filter(!is.na(seafood_yn))
nrow(labeled) + nrow(unlabeled) == nrow(df)

###### create training and test with the same prob #######

#create index of stratified 80:20 split that maintain proportion of label.
index <- caret::createDataPartition(labeled$seafood_yn, times = 1,
                                    p = 0.8, list= FALSE)

# train and test
train <- labeled[index,]
test <- labeled[-index,]
```

Next, we will create document frequency matrix from datasets so we can count each unique number of words in unlabeled dataset, dummy train dataset, and dummy test dataset.

```
#################### add vocabulary ###############
#create a corpus for unlabeled dataset
corpus <- corpus(unlabeled$translatedText)

#create document frequency matrix with verb
dfm.unlabeled <- dfm(corpus,
              tolower=TRUE,
              remove_punct =TRUE,
              remove_numbers=TRUE,
              remove_symbols = TRUE,
              remove_hyphens = TRUE,
              remove=stopwords('english', source = "snowball"),
              stem=TRUE,
              verbose=TRUE)


############ create a corpus for train  dataset #######
#create a corpus for unlabeled dataset
corpus <- corpus(train$translatedText)

#create document frequency matrix with verb
dfm.train <- dfm(corpus,
```

```
              tolower=TRUE,
              remove_punct =TRUE,
              remove_numbers=TRUE,
              remove_symbols = TRUE,
              remove_hyphens = TRUE,
              remove=stopwords('english', source = "snowball"),
              stem=TRUE,
              verbose=TRUE)


############ create a corpus for test  dataset #######
corpus <- corpus(test$translatedText)

#create document frequency matrix with verb
dfm.test <- dfm(corpus,
              tolower=TRUE,
              remove_punct =TRUE,
              remove_numbers=TRUE,
              remove_symbols = TRUE,
              remove_hyphens = TRUE,
              remove=stopwords('english', source = "snowball"),
              stem=TRUE,
              verbose=TRUE)
```

Once we had all three document frequency, we make a table that contain words count from each matrix

```
######### table that contain all word count in unlabeled ########
dfm.unlabeled.trimmed <- dfm_trim(dfm.unlabeled, min_docfreq = 15, verbose = TRUE)
dim(dfm.unlabeled.trimmed)
count.unlabeled <- apply(dfm.unlabeled.trimmed[1:100000,], 2, sum)
count.unlabeled2 <- apply(dfm.unlabeled.trimmed[100001:200000,], 2, sum)
count.unlabeled3 <- apply(dfm.unlabeled.trimmed[200001:300000,], 2, sum)
count.unlabeled4 <- apply(dfm.unlabeled.trimmed[300001:392095,], 2, sum)
count.unlabeled <- count.unlabeled+count.unlabeled2+count.unlabeled3+count.unlabeled4
count.unlabeled <- data.frame(name = names(count.unlabeled),
            freq = count.unlabeled)

######### table that contain all word count in dummy train ########
count.train <- apply(dfm.train, 2, sum)
count.train <- data.frame(name = names(count.train),
            freq = count.train)
######### table that contain all word count in dummy test ########
count.test <- apply(dfm.test,2,sum)
count.test <- data.frame(name = names(count.test),
            freq = count.test)
```

There might be too much data to evaluate. To make it feasible to add more sample we decided to choose 10,000 unique words that are most frequently appear. Then, we join top term from three count table that we had before. The results is a table of most frequent data that are not available in train and/or test dataset.

```
## top words
topterm.train <- count.train %>%
  arrange(desc(freq)) %>%
```

```
  slice(1:10000)

topterm.test <- count.test %>%
  arrange(desc(freq)) %>%
  slice(1:1000)

topterm.unlabeled <- count.unlabeled %>%
  arrange(desc(freq)) %>%
  slice(1:3000)

nothing.on.labeled <- topterm.unlabeled %>%
  left_join(topterm.train, by = "name") %>%
  left_join(topterm.test, by = "name") %>%
  filter(is.na(freq.y) & is.na(freq))
```

Based on table above, we decide to take 760 observations of words that aren't available in labeled dataset.

```
nothing.on.labeled.smp <- nothing.on.labeled %>%
  mutate(freq.x = round(freq.x/100)+1)

sampleoct22 <- vector(mode = "list", length = nrow(nothing.on.labeled))
for (i in seq_along(sampleoct22)) {
  sampleoct22[[i]] <- df.unlabeled %>%
    filter(str_detect(translatedText,nothing.on.labeled$name[i])) %>%
    select(id, first_appeared) %>%
    arrange(desc(first_appeared)) %>%
    slice(1:nothing.on.labeled.smp$freq.x[[i]]) %>% ##take number of dish from latest dish (we low on n
    select(id)
}

sampleoct22 <- do.call(rbind, sampleoct22)
sampleoct22 <- unique(sampleoct22)
sampleoct22.vector <- sampleoct22$id

## add more label for pattern that are not available in training (see steps on appendix)
## our pattern
new.labeled.22oct <- df.unlabeled %>%
  filter(id %in% sampleoct22.vector)
write.csv(new.labeled.22oct, "newlabeled22oct.csv")
```

Then, we labele it manually. It worth to mention that even though we already translate dish name to english, one dish name are remain unidentifiable whether it is a seafood or non seafood. We decided to leave it NA.

```
newlabeled22oct <- read_csv("newlabeled22oct.csv")
newlabeled22oct <- newlabeled22oct[,-1]
which(is.na(newlabeled22oct$seafood_yn))
newlabeled22oct <- newlabeled22oct[-598,]
```

At last, we add our new sample to our labeled dataset. Then we remove it from unlabeled dataset. Finally we bind them together in df.all that contain all dataset.

```r
##join it to labeled
labeled <- rbind(labeled, newlabeled22oct)

## remove it from unlabeled
unlabeled <- unlabeled %>%
  anti_join(newlabeled22oct, by = "id")

## join it to make one single datasets
df.all <- rbind(unlabeled, labeled)

## check
nrow(df.all) == nrow(df)

## save
write_csv(df.all, file = "dfalltranslated.csv")
write_csv(labeled, file = "dflabeledtranslated.csv")
write_csv(unlabeled, file = "dfunlabeledtranslated.csv")
rm(list = ls())
```

It worth to mention that There are some human error in labeled dataset, especially when dish name is not in english. Therefore, we still continue to evaluate seafood_yn variable in labeled dataset manually. Our final labeled dataset after final evaluation is dflabeledtranslated_edited.csv.